

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»  
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ  
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## **ПОЯСНЮВАЛЬНА ЗАПИСКА**

до кваліфікаційної роботи освітнього ступеня «бакалавр»  
за спеціальністю 121 «Інженерія програмного забезпечення»  
(освітня програма «Інженерія програмного забезпечення»)

на тему:

**«Оптимізація фінансових процесів у CRM-системі для  
шкіл за допомогою автоматизованих рішень»**

Виконав студент групи ПЗ-21-5  
САДОЛІНСЬКИЙ Віталій Анатолійович

Керівник роботи:  
ГРОМСЬКИЙ Олександр Олександрович

Рецензент:  
ЧИЖМОТРЯ Олена Геннадіївна

Житомир – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»  
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ  
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

**«ЗАТВЕРДЖУЮ»**  
Зав. кафедри інженерії  
програмного забезпечення  
Тетяна Вакалюк  
**«26» лютого 2024 р.**

**ЗАВДАННЯ**  
**на кваліфікаційну роботу**

Здобувач вищої освіти: **САДОЛІНСЬКИЙ Віталій Анатолійович**

Керівник роботи: **ГРОМСЬКИЙ О.О.**

Тема роботи: **«Оптимізація фінансових процесів у CRM-системі для шкіл за допомогою автоматизованих рішень»,**

затверджена Наказом закладу вищої освіти від **«14» лютого 2025 р., №61/с**

Вихідні дані для роботи: **фінансові процеси у приватних школах, зокрема облік оплат, звітність та інтеграція CRM-систем із банківськими сервісами.**

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

| Розділ | Консультант    | Завдання видав | Завдання прийняв |
|--------|----------------|----------------|------------------|
| 1      | Громський О.О. | 27.02.2025р.   | 27.02.2025р.     |
| 2      | Громський О.О. | 27.02.2025р.   | 27.02.2025р.     |
| 3      | Громський О.О. | 27.02.2025р.   | 27.02.2025р.     |

## РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з CRM-системи для шкіл та пояснювальної записки. Пояснювальна записка до випускної роботи містить 81 сторінок, 39 ілюстрацію, 18 таблиць. У роботі використано 36 джерел літератури.

Метою роботи є оптимізація фінансових процесів у навчальних закладах шляхом розробки CRM-системи з автоматизованими рішеннями для обліку оплат, синхронізації з банківськими сервісами та формування звітності.

У процесі реалізації використано мову програмування Java та фреймворк Spring Boot. Автоматизовано облік платежів, впроваджено інтеграцію з API ПриватБанку, а також створено систему формування фінансових звітів. Отримані результати дозволяють суттєво знизити навантаження на бухгалтерію та підвищити прозорість фінансових операцій у школі.

Результати дослідження можуть бути впроваджені в приватних освітніх установах для покращення фінансового менеджменту.

**КЛЮЧОВІ СЛОВА:** CRM, JAVA, ФІНАНСИ, ПРИВАТ БАНК, ШКОЛА.

|           |      |                  |        |      |                                                           |      |         |
|-----------|------|------------------|--------|------|-----------------------------------------------------------|------|---------|
|           |      |                  |        |      | «Житомирська політехніка».25.121.11.000                   |      |         |
|           |      |                  |        |      |                                                           |      |         |
| Змн.      | Арк. | № докум.         | Підпис | Дата | Пояснювальна записка для кваліфікаційної дипломної роботи |      |         |
| Розроб.   |      | Садолінський В.А |        |      |                                                           |      |         |
| Перевір.  |      | Громський О.О.   |        |      |                                                           |      |         |
| Керівник  |      |                  |        |      |                                                           |      |         |
| Н. контр. |      |                  |        |      |                                                           |      |         |
| Зав. каф. |      |                  |        |      | ІПЗ.КР.Б-121-5-ПЗ                                         |      |         |
|           |      |                  |        |      | Літ.                                                      | Арк. | Аркушів |
|           |      |                  |        |      |                                                           | 3    | 81      |

## ABSTRACT

The qualification work consists of a CRM system for private schools and an explanatory note. The explanatory note contains 81 pages, 39 illustrations, 18 tables. The work uses 36 literary sources.

The aim of the thesis is to optimize financial processes in educational institutions by developing a CRM system with automated solutions for payment management, synchronization with banking services, and generation of financial reports.

The implementation was carried out using the Java programming language and the Spring Boot framework. The system automates tuition payment tracking, integrates with PrivatBank's API, and generates financial reports. The achieved results significantly reduce the workload of accounting staff and increase the transparency of financial transactions at the school.

The results of the study can be implemented in private educational institutions to improve the efficiency of financial management.

**KEYWORDS:** CRM, JAVA, FINANCE, PRIVATBANK, SCHOOL

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         | 4    |

## ЗМІСТ

|                                                                                                                     |           |
|---------------------------------------------------------------------------------------------------------------------|-----------|
| <b>РЕФЕРАТ .....</b>                                                                                                | <b>3</b>  |
| <b>ABSTRACT .....</b>                                                                                               | <b>4</b>  |
| <b>ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ .....</b>                                                                              | <b>6</b>  |
| <b>ВСТУП.....</b>                                                                                                   | <b>7</b>  |
| <b>РОЗДІЛ 1. АНАЛІЗ СУЧАСНИХ ПІДХОДІВ ДО АВТОМАТИЗАЦІЇ<br/>ФІНАНСОВИХ ПРОЦЕСІВ В ОСВІТНІХ CRM-СИСТЕМАХ .....</b>    | <b>9</b>  |
| 1.1. Постановка задачі.....                                                                                         | 9         |
| 1.2. Аналіз аналогів програмного продукту. ....                                                                     | 13        |
| 1.3. Обґрунтування вибору інструментальних засобів та вимоги до апаратного<br>забезпечення.....                     | 17        |
| Висновки до першого розділу.....                                                                                    | 22        |
| <b>РОЗДІЛ 2. ПРОЕКТУВАННЯ CRM-СИСТЕМИ ДЛЯ ОПТИМІЗАЦІЇ<br/>ФІНАНСОВИХ ПРОЦЕСІВ У ШКОЛАХ .....</b>                    | <b>23</b> |
| 2.1. Визначення варіантів використання та об'єктно-орієнтованої структури<br>системи. ....                          | 23        |
| 2.2. Розробка бази даних.....                                                                                       | 28        |
| 2.3. Реалізація системи.....                                                                                        | 35        |
| Висновки до другого розділу. ....                                                                                   | 40        |
| <b>РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З CRM-СИСТЕМОЮ ДЛЯ<br/>АВТОМАТИЗАЦІЇ ФІНАНСОВИХ ПРОЦЕСІВ У ШКОЛІ .....</b> | <b>41</b> |
| 3.1. Структура інтерфейсу системи та тестування системи.....                                                        | 41        |
| 3.2. Сценарії використання системи користувачами різних ролей.....                                                  | 74        |
| Висновки до третього розділу.....                                                                                   | 76        |
| <b>ВИСНОВКИ .....</b>                                                                                               | <b>77</b> |
| <b>ЛІТЕРАТУРНІ ДЖЕРЕЛА.....</b>                                                                                     | <b>78</b> |
| <b>ДОДАТКИ.....</b>                                                                                                 | <b>81</b> |

## ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API - прикладний програмний інтерфейс.

БД - база даних.

CRM – система управління взаємовідносинами з клієнтами (Customer Relationship Management).

ПЗ - програмне забезпечення.

СУБД - система управління базами даних.

SDK – набір інструментів для розробки програмного забезпечення (Software Development Kit).

MVC - шаблон проєктування «Модель–Подання–Контролер» (Model-View-Controller).

CRUD - базові операції з даними: створення, читання, оновлення, видалення (Create, Read, Update, Delete).

REST - архітектурний стиль взаємодії між клієнтом і сервером у вебсередовищі (Representational State Transfer).

JSON - текстовий формат обміну даними (JavaScript Object Notation).

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         | 6    |

## ВСТУП

**Актуальністю теми** є сучасні освітні заклади, які дедалі більше потребують ефективних цифрових інструментів для автоматизації фінансових процесів. Значна частина адміністративного навантаження пов'язана з веденням обліку оплат, складанням звітності та контролем надходжень. У традиційних умовах ці процеси здебільшого виконуються вручну або з використанням застарілого програмного забезпечення, що часто призводить до помилок, затримок та низької прозорості.

В умовах цифрової трансформації освітнього середовища актуальним є створення CRM-систем, які не лише забезпечують зручну взаємодію з батьками та учнями, але й інтегрують функції фінансового менеджменту. Використання автоматизованих рішень дозволяє зменшити обсяг рутинної роботи, забезпечити своєчасний контроль оплат та створити надійні аналітичні інструменти.

Особливої значущості набуває інтеграція таких систем із банківськими сервісами, зокрема API ПриватБанку, що дозволяє в режимі реального часу отримувати та обробляти платіжні дані. Такий підхід сприяє підвищенню фінансової прозорості в навчальному закладі, зменшує ризики людського чинника та оптимізує роботу бухгалтерії.

**Метою** випускної кваліфікаційної роботи є розробка CRM-системи для оптимізації фінансових процесів за допомогою автоматизованих рішень.

Для реалізації поставленої мети нам потрібно виконати наступні завдання:

1. Проаналізувати фінансові процеси в освітніх установах, зокрема системи обліку оплат, підготовки звітності та інтеграції з банківськими сервісами.
2. Провести огляд існуючих CRM-систем, що використовуються в навчальних закладах, та виявити їх недоліки у сфері фінансового менеджменту.
3. Обрати інструменти та засоби розробки, зокрема архітектурні підходи та сторонні сервіси.

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         | 7    |

4. Розробити CRM-систему з функціональністю обліку оплат, формування фінансових звітів і синхронізації з банківськими системами.

5. Протестувати систему в умовах реального використання, та оцінити ефективність впроваджених автоматизованих рішень.

6. Сформулювати рекомендації щодо подальшого використання та масштабування розробленої системи в інших освітніх установах.

**Об’єктом дослідження** виступають фінансові процеси, що відбуваються в рамках діяльності освітніх установ, зокрема в приватних школах. До таких процесів належать облік та контроль за платежами за навчання, обробка фінансових транзакцій, управління залишками коштів, повернення оплат, а також формування актуальної звітності для адміністрації, бухгалтерії, викладачів та інших зацікавлених сторін. Окрему увагу в межах дослідження приділено способам взаємодії CRM-систем із зовнішніми фінансовими інструментами, зокрема банківськими сервісами, що дозволяють автоматизувати процес надходження та обліку платежів.

**Предметом дослідження** є сукупність методів, інструментів і технологій, що забезпечують ефективну автоматизацію вищезазначених фінансових процесів у межах CRM-систем, орієнтованих на потреби шкільних адміністрацій. Це охоплює реалізацію інтеграцій з API банківських установ, побудову гнучких механізмів збереження й обробки інформації про платежі, налаштування ставок та тарифів, облік заробітної плати викладачам, а також створення достовірної, аналітичної та прозорої фінансової звітності. У центрі предметної області знаходяться програмні рішення, які дозволяють зменшити вплив людського чинника, скоротити витрати часу на облік та контроль, забезпечити прозорість і підвищити точність усіх фінансових операцій у навчальних закладах.

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         | 8    |

# РОЗДІЛ 1. АНАЛІЗ СУЧАСНИХ ПІДХОДІВ ДО АВТОМАТИЗАЦІЇ ФІНАНСОВИХ ПРОЦЕСІВ В ОСВІТНІХ CRM-СИСТЕМАХ

## 1.1. Постановка задачі.

Метою кваліфікаційної роботи є розробка CRM-системи для освітніх установ з використанням сучасних інформаційних технологій та інструментів автоматизації, що забезпечить оптимізацію фінансових процесів, зокрема обліку оплат, формування звітності та інтеграції з банківськими сервісами. Реалізація такого рішення сприятиме підвищенню прозорості, ефективності та зменшенню навантаження на фінансовий персонал навчальних закладів.

В умовах цифровізації освіти та зростання кількості освітніх закладів, значно збільшується обсяг фінансових операцій, що потребують автоматизації. Традиційні методи ведення фінансового обліку (Excel, ручне внесення даних, паперові звіти) не відповідають вимогам сучасного управління, оскільки є вразливими до помилок, дублювання інформації. У зв'язку з цим, впровадження CRM-системи, яка автоматизує всі етапи фінансових процесів, є не лише доцільним, а й критично важливим для ефективного функціонування навчального закладу.

CRM-система (Customer Relationship Management) для освітніх закладів - це програмне рішення, призначене для управління взаємодією між адміністрацією школи, фінансовими відділами, учнями та їхніми батьками. Основна мета впровадження такої системи - забезпечити централізований облік даних, автоматизувати фінансові процеси, зменшити вплив людського фактора та підвищити прозорість фінансових операцій.

У сучасних умовах школи активно розширюють перелік внутрішніх бізнес-процесів, серед яких особливу увагу приділено обліку платежів, формуванню фінансової звітності, нагадуванням про заборгованість та інтеграції з банківськими сервісами. Саме тому постає потреба у впровадженні CRM-системи з потужним модулем фінансового обліку.

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         | 9    |

CRM-система розробляється з використанням мікросервісної архітектури. Застосування такої архітектури надає низку суттєвих переваг, які безпосередньо впливають на гнучкість, масштабованість та надійність програмного забезпечення.

По-перше, мікросервісний підхід дозволяє розділити систему на окремі незалежні компоненти, кожен з яких відповідає за конкретну бізнес-функцію (обробка платежів, інтеграція з банками, формування звітів, сповіщення тощо). Такий поділ забезпечує можливість незалежної розробки, тестування та розгортання кожного мікросервісу без необхідності зупинки всієї системи. Це значно спрощує супровід та вдосконалення окремих частин ПЗ.

По-друге, система стає більш масштабованою. Наприклад, у разі збільшення навантаження на певний модуль, його можна масштабувати окремо, не впливаючи на інші компоненти системи. Це дозволяє оптимально використовувати ресурси інфраструктури та підтримувати стабільну роботу системи за високого навантаження.

По-третє, мікросервіси забезпечують відмовостійкість: збій в одному з сервісів не призводить до зупинки всієї системи. Це особливо важливо в контексті обробки фінансових операцій, де надійність і безперервність роботи мають критичне значення.

Крім того, мікросервісна архітектура полегшує адаптацію та масштабування системи під потреби конкретного навчального закладу. У разі потреби можливо швидко додати новий функціональний модуль або інтегрувати систему з іншими зовнішніми сервісами без серйозного втручання в наявну структуру.

Система буде складатися з наступних основних мікро-сервісів:

- Головний мікро-сервіс (Root) - обробка платежів, розподілення коштів та інші базові фінансові операції (повернення коштів і тд);
- Мікросервіс інтеграції з банківськими системами (Integration) - забезпечує інтеграцію з API банків (зокрема ПриватБанку) для автоматичної обробки транзакцій;

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         | 10   |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         |      |

- Мікросервіс фінансової звітності (Report) - створює аналітичні та бухгалтерські звіти;
- Мікросервіс сповіщень (Notification) - надсилає нагадування про оплату, заборгованість, або інші важливі повідомлення батькам та адміністрації.

Завдяки поділу функціоналу на мікросервіси, система може легко масштабуватись, оновлюватись без зупинки всієї платформи, а також адаптуватись до вимог замовника.

У результаті має бути створено CRM-систему, призначену для автоматизації фінансових процесів у приватних освітніх закладах. Основною метою є полегшення ведення обліку оплат за навчання, формування фінансової звітності та інтеграція з банківськими системами, що забезпечить оперативну обробку транзакцій.

Система має бути зручною для використання як працівниками навчального закладу, так і студентами та батьками студентів. Передбачається використання мікросервісної архітектури, яка дозволить забезпечити гнучке масштабування, надійність і простоту обслуговування кожного окремого компонента.

Основні етапи розробки системи включають:

#### 1. Проєктування архітектури системи:

- формування загальної структури взаємодії мікросервісів;
- визначення зон відповідальності кожного сервісу (платежі, інтеграція, звітність, сповіщення);
- розробка схем взаємодії через REST API та системи маршрутизації запитів;
- проєктування схеми баз даних для кожного мікросервісу.

#### 2. Розробка серверної частини на основі Spring Boot:

- реалізація головного мікросервісу (Root Service), що відповідає за обробку платежів, фінансові транзакції, повернення коштів тощо;
- створення мікросервісу інтеграції з банками (Integration Service);

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         | 11   |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         |      |

- розробка сервісу формування звітності (Report Service) для генерації фінансових аналітичних і бухгалтерських звітів;
- реалізація сервісу сповіщень (Notification Service) для надсилання нагадувань та повідомлень користувачам;
- забезпечення взаємодії мікросервісів через внутрішню API-шину та брокери повідомлень Kafka, Pub/Sub, RabbitMQ.

### 3. Розробка системи авторизації:

- впровадження механізмів безпечного входу користувачів із розмежуванням прав доступу;
- інтеграція JWT (JSON Web Token) для збереження сесій і перевірки прав доступу до ресурсів;
- реалізація централізованого сервісу авторизації (Security Service), спільного для всієї системи.

### 4. Реалізація бізнес-логіки системи:

- створення функціоналу реєстрації/перегляду оплат за навчання;
- реалізація логіки створення, фільтрації та експорту звітів по філіях, групах та часових періодах;
- інтеграція можливостей налаштувань (наприклад, типи послуг, валюти, шаблони повідомлень тощо).

### 5. Тестування та вдосконалення системи:

- перевірка банківської інтеграції, включно з реальними тестовими транзакціями;
- тестування авторизації, навантаження та безпеки (включаючи захист кінцевих точок API);
- тестування зручності інтерфейсів для різних категорій користувачів.

Результатом стане повнофункціональна CRM-система для приватних шкіл, яка дозволяє автоматизувати ключові фінансові операції, мінімізувати навантаження на бухгалтерський персонал та забезпечити прозоре управління фінансами в навчальному закладі.

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         | 12   |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         |      |

## 1.2. Аналіз аналогів програмного продукту.

У межах аналізу було розглянуто низку існуючих CRM-систем, орієнтованих на навчальні заклади або маючих функціональні модулі для автоматизації фінансового обліку. До таких систем належать: Schoology, BSchool CRM, SimpleSchool. Основною метою аналізу є виявлення їх можливостей та обмежень, що дозволить обґрунтувати доцільність розробки нової системи.

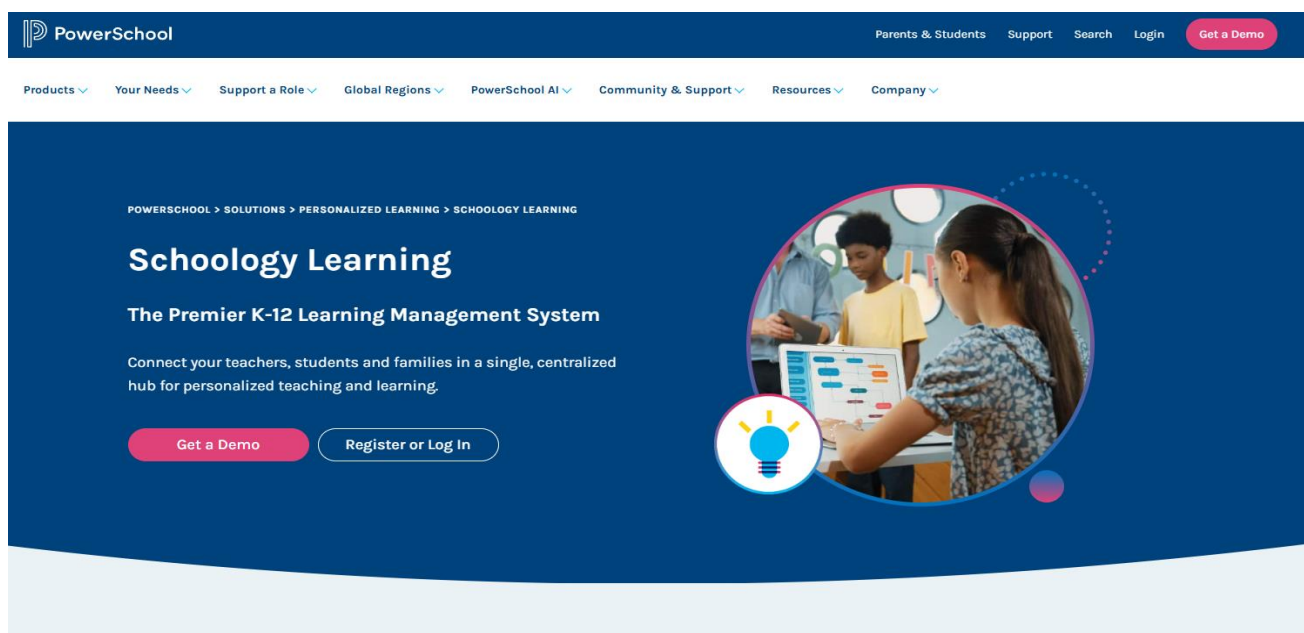


Рис.1.1. CRM-система Schoology.

Schoology - популярна платформа для організації дистанційного навчання, яка має базові CRM-функції: ведення учнівських профілів, комунікація між учнями та викладачами, управління розкладом.

Переваги:

- Інтуїтивно зрозумілий інтерфейс;
- Потужні інструменти для організації навчального процесу;
- Підтримка інтеграцій з LMS та сторонніми сервісами;
- Розвинена система комунікації між учнями і викладачами.

Недоліки:

- Відсутній фінансовий модуль;
- Неможливість відстеження оплат та автоматичного формування звітів;

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         | 13   |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         |      |

- Не адаптована до українського ринку та локального законодавства;
- Орієнтація лише на навчальний контент, а не на адміністрування школи.



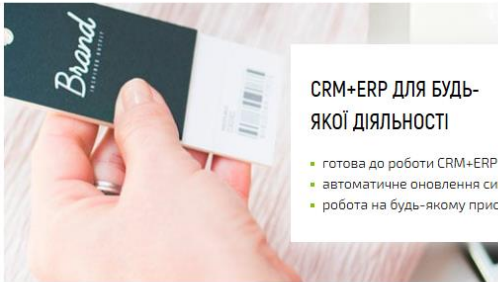
МОЖЛИВОСТІКОРІБКАХМАРВАРТІСТЬПОСЛУГИПАРТНЕРИМАГАЗИНДОКУМЕНТАЦІЯПРО НАСКОНТАКТИ

0 800 755 221

ЗАМОВИТИ

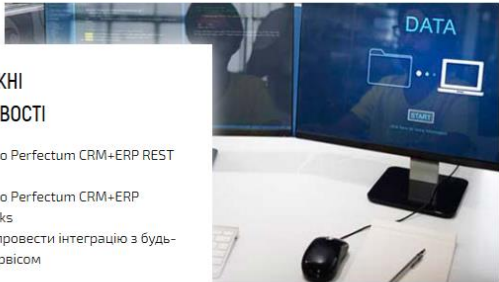
УКР

PERFECTUM CRM+ERP- ЦЕ...



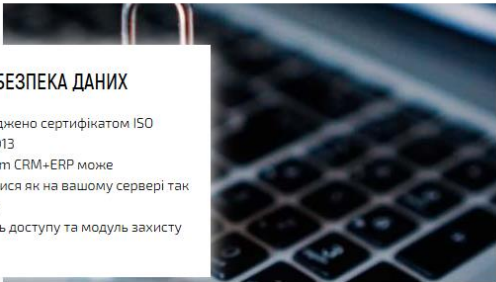
CRM+ERP ДЛЯ БУДЬ-ЯКОЇ ДІЯЛЬНОСТІ

- готова до роботи CRM+ERP
- автоматичне оновлення системи
- робота на будь-якому пристрої



БЕЗМЕЖНІ МОЖЛИВОСТІ

- доступно Perfectum CRM+ERP REST API
- доступно Perfectum CRM+ERP Webhooks
- можна провести інтеграцію з будь-яким сервісом



ПОВНА БЕЗПЕКА ДАНИХ

- Підтверджено сертифікатом ISO 27001:2013
- Perfectum CRM+ERP може зберігатися як на вашому сервері так і у хмарі
- контроль доступу та модуль захисту по IP



ПРОДУКТ ВІДЗНАЧЕНИЙ НАГОРОДАМИ

- Премія «Вибір Країни»
- Українська народна премія
- Українська CRM/ERP-система 2023 року
- CRM система 2022 року
- та ще більше 15 інших

Рис.1.2. CRM-система Perfectum CRM.

Perfectum CRM - це потужна українська CRM-система, орієнтована на комплексну автоматизацію бізнес-процесів. Вона має галузеві рішення, зокрема для освітніх установ. Для приватних шкіл Perfectum CRM дозволяє автоматизувати фінансові процеси (платежі, борги, звітність), організовувати комунікацію з батьками, вести облік навчальних груп, абонементів та відвідуваності.

#### Переваги:

- Повна автоматизація шкільних процесів, включаючи фінансовий облік, управління персоналом та документообіг.
- Онлайн-доступ через будь-який браузер, що забезпечує зручність використання.
- Можливість адаптації під специфічні потреби навчального закладу.

|      |      |                   |        |      |                                         |      |
|------|------|-------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А. |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.    |        |      |                                         |      |
| Змн. | Арк. | № докум.          | Підпис | Дата |                                         | 14   |

Недоліки:

- Може бути складною для впровадження без технічної підтримки.
- Вартість впровадження та обслуговування може бути високою для невеликих шкіл.

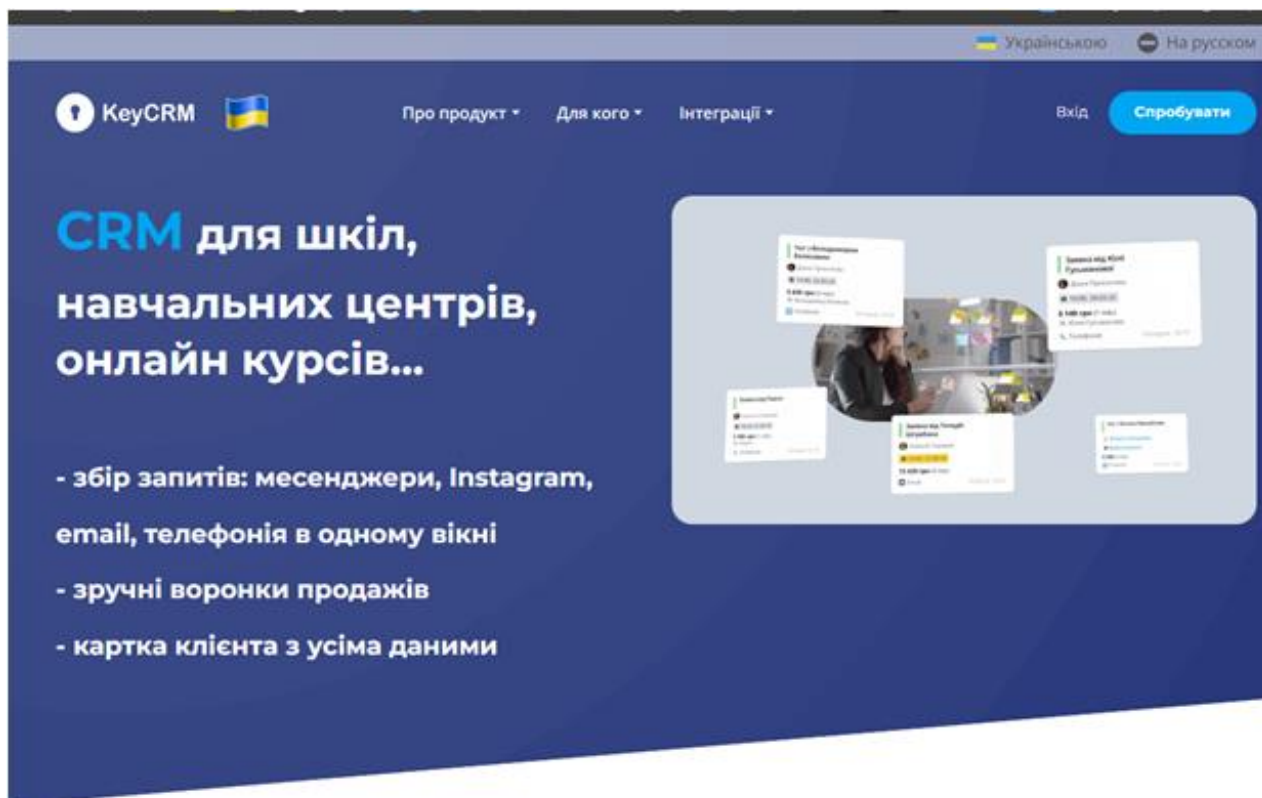


Рис.1.3. CRM-система KeyCRM.

KeyCRM - це хмарна CRM-система, створена переважно для малого та середнього бізнесу. Вона також адаптується для роботи в освітніх закладах, зокрема у сфері онлайн-навчання, репетиторських центрів та приватних шкіл

Переваги:

- Інтеграція з популярними месенджерами та платіжними системами.
- Дружній інтерфейс, що не потребує спеціального навчання.
- Можливість самостійного налаштування без залучення програмістів.

Недоліки:

- Обмежений функціонал для великих освітніх установ з комплексними потребами.
- Може не підтримувати специфічні модулі, необхідні для деяких шкіл.

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         | 15   |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         |      |

З урахуванням виявлених переваг та недоліків сформуємо порівняльну таблицю зазначених програмних рішень (табл. 1.1).

Таблиця 1.1.

Таблиця порівняння CRM систем.

| Характеристика / Платформа     | KeyCRM                                    | Perfectum CRM                                  | Schoology                                         |
|--------------------------------|-------------------------------------------|------------------------------------------------|---------------------------------------------------|
| Призначення                    | Бізнес-орієнтована CRM                    | Універсальна CRM з освітнім модулем            | Освітня LMS-платформа                             |
| Фінансовий облік               | Базова функціональність                   | Розширений функціонал: оплати, борги, звіти    | Відсутній або реалізується через сторонні сервіси |
| Інтеграція з банками           | Обмежена (через API або сторонні сервіси) | Пряма інтеграція з банками, зокрема ПриватБанк | Не передбачена                                    |
| Управління навчальним процесом | Обмежене                                  | Можлива адаптація під школу                    | Основна спеціалізація                             |
| Повідомлення та сповіщення     | Email, месенджери                         | Email, SMS, Telegram                           | Вбудовані повідомлення                            |
| Звіти та аналітика             | Стандартні бізнес-звіти                   | Розширені фінансові звіти                      | Звіти про активність учнів                        |
| Можливість кастомізації        | Обмежена                                  | Висока                                         | Середня                                           |
| Інтерфейс українською мовою    | Так                                       | Так                                            | Частково (не повністю локалізована)               |
| Ціна / ліцензія                | SaaS, підписка                            | SaaS                                           | SaaS, є безкоштовний тариф                        |
| Підтримка ролей користувачів   | Менеджери, клієнти                        | Адміністрація, бухгалтери, вчителі, батьки     | Учні, вчителі, адміністратори                     |

|          |                           |                                           |                                     |
|----------|---------------------------|-------------------------------------------|-------------------------------------|
| Переваги | Простота використання     | Комплексність, гнучка інтеграція          | Ідеальна для дистанційного навчання |
| Недоліки | Не пристосована до освіти | Потребує налаштування під конкретну школу | Відсутність фінансових модулів      |

Аналіз показав, що на ринку існують окремі рішення для управління навчальним процесом та CRM-системи для освітніх установ, однак більшість із них мають обмежену функціональність у сфері автоматизованого фінансового менеджменту, а також не передбачають інтеграції з українськими банківськими сервісами та гнучкої генерації звітності.

Розроблювана CRM-система буде вирізнятися низкою нововведень:

- інтеграція з API ПриватБанку для автоматичного отримання платіжних даних;
- автоматизація рутинних процесів: створення звітів, облік оплат, нагадування користувачам;
- реалізація архітектури, що забезпечує масштабованість і гнучкість;
- модульність, що дозволяє легко адаптувати систему під індивідуальні потреби навчального закладу.

Таким чином, запропоноване рішення не лише покриває слабкі сторони існуючих аналогів, а й запроваджує новий підхід до фінансового управління в приватних школах з акцентом на прозорість, ефективність і автоматизацію.

### 1.3. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення.

Під час реалізації програмного забезпечення було обрано набір інструментальних засобів, що відповідає сучасним стандартам розробки, забезпечує масштабованість, модульність і надійність системи, а також полегшує процес розробки та підтримки застосунку.

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         | 17   |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         |      |

Для серверної частини обрано Java як основну мову програмування, оскільки вона є однією з найбільш стабільних, безпечних та підтримуваних мов для розробки корпоративних систем. Основу серверної логіки становить Spring Boot, який надає широкий набір функцій для побудови REST API, реалізації безпеки, обробки запитів, керування залежностями, роботи з базами даних тощо. Його модульність та велика екосистема дає можливість швидко інтегрувати нові компоненти, масштабувати проєкт та підтримувати стабільну структуру коду.

Spring Boot також забезпечує легку інтеграцію з іншими компонентами екосистеми Spring, зокрема Spring Security, Spring Data, Spring Cloud та іншими. Для підтвердження доцільності використання Spring Boot було проведено порівняльний аналіз з іншими популярними фреймворками для розробки бекенду, а саме Quarkus, Micronaut та Jakarta EE (табл. 1.2).

Таблиця 1.2

#### Порівняння Java-фреймворків для розробки серверної частини.

| Критерій               | Spring Boot                | Quarkus               | Micronaut | Jakarta EE      |
|------------------------|----------------------------|-----------------------|-----------|-----------------|
| Продуктивність         | Висока                     | Дуже висока (GraalVM) | Висока    | Середня         |
| Швидкість запуску      | Середня                    | Висока                | Висока    | Низька          |
| Крива навчання         | Помірна                    | Помірна               | Висока    | Висока          |
| Спільнота/Документація | Дуже велика                | Зростаюча             | Помірна   | Стабільна       |
| Підтримка екосистеми   | Повна (Spring Cloud, Data) | Обмежена              | Обмежена  | Стандартизована |
| Інтеграція з DB        | Spring Data, JPA           | Panache, Hibernate    | JPA       | JPA             |
| Мікросервіси           | Підтримується повністю     | Так                   | Так       | Обмежено        |

Було реалізовано мікросервісну архітектуру, яка дозволяє логічно розділити систему на окремі незалежні сервіси. Кожен мікросервіс відповідає за окремий бізнес-процес, наприклад, обробку платежів, управління користувачами, звітність тощо. Такий підхід забезпечує високий рівень ізоляції між компонентами, полегшує масштабування системи під час зростання навантаження, а також підвищує надійність, адже збої в одному сервісі не впливають на роботу інших.

Для побудови та розгортання кожного мікросервісів використовуються Docker-контейнери, що забезпечують ізольоване середовище для виконання, уніфікують інфраструктуру та пришвидшують процеси деплою. Кожен сервіс контейнеризується окремо з урахуванням його залежностей. Це дозволяє легко відтворити середовище у різних етапах розробки.

GitHub Actions використовується як платформа для реалізації CI/CD-процесів. Автоматизація включає перевірку коду, збірку, виконання юніт-тестів, створення Docker-образів та деплой до хмарної інфраструктури. Таким чином забезпечується висока швидкість і надійність оновлення ПЗ без простою системи.

Мікросервісна архітектура дозволяє також реалізовувати CI/CD-процеси для кожного сервісу окремо, що підвищує гнучкість та швидкість оновлення. Для комунікації між сервісами використовується OpenFeign, який спрощує виклики до зовнішніх або внутрішніх API шляхом опису лише інтерфейсів.

Для зберігання даних використовується PostgreSQL - потужна, об'єктно-реляційна система управління базами даних з відкритим кодом. PostgreSQL забезпечує високу продуктивність, надійність та підтримує розширені механізми безпеки, транзакційність, ACID-сумісність і реплікацію. [7] Завдяки підтримці складних типів даних, індексації, тригерів та збережених процедур, PostgreSQL є ідеальним вибором для побудови складних бізнес-додатків, зокрема таких, як CRM для управління фінансами в освітніх установах.

Контроль версій бази даних здійснюється за допомогою Flyway - інструмента, який автоматизує міграції схеми бази даних. Це дозволяє зберігати консистентність структури БД між середовищами розробки, тестування та продакшену, уникати людських помилок і забезпечити просте відтворення будь-якої версії структури БД.

MapStruct застосовується для автоматизованого мапінгу між DTO (об'єктами передачі даних) і сутностями. Це дозволяє зменшити обсяг шаблонного коду та уникнути помилок, пов'язаних з ручним копіюванням даних між шарами застосунку. Завдяки використанню анотацій, MapStruct генерує

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         | 19   |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         |      |

мапінг-код під час компіляції, що гарантує високу продуктивність і легкість у супроводі.

Також у проєкті застосовується Lombok - бібліотека, що дозволяє зменшити шаблонний код у класах, додаючи гетери, сетери, конструктори, методи toString() тощо за допомогою простих анотацій. Це підвищує читабельність коду і пришвидшує процес розробки.

Для контролю змін у сутностях використовується Hibernate Envers. Він дозволяє автоматично зберігати історію змін у таблицях бази даних. Для більш детального відстеження фінансових змін, повернень та інших подій була розроблена власна система відстеження змін в сутностях.

Spring Security і JWT використовуються для забезпечення авторизації користувачів. Spring Security дозволяє налаштовувати правила доступу на основі ролей, а JWT (JSON Web Token) - безпечно передавати інформацію про користувача між клієнтом і сервером без необхідності зберігати сесії на сервері.

Google Cloud Storage (GCS) було обрано як хмарне сховище для зберігання файлів, таких як чеки, документи, фінансові звіти та інші медіа. Його масштабованість і висока доступність дозволяє забезпечити безперервний доступ до даних з будь-якої точки світу, що важливо для навчальних закладів.

Вся архітектура проєкту побудована на основі REST API, що забезпечує зручну, стандартизовану взаємодію як між фронтендом і бекендом, так і між окремими мікросервісами системи. REST (Representational State Transfer) дозволяє використовувати зрозумілі HTTP-методи для виконання CRUD-операцій, що значно спрощує розробку, тестування та підтримку функціоналу.

Завдяки незалежності REST-ендпоінтів та використанню формату обміну даними JSON, досягається висока гнучкість при розширенні функціональності: нові модулі можуть бути додані без порушення роботи існуючих компонентів.

Такий підхід дозволяє ефективно масштабувати систему - як вертикально (покращуючи потужність окремих сервісів), так і горизонтально (шляхом розгортання додаткових екземплярів мікросервісів).

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         | 20   |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         |      |

Крім того, REST API полегшує інтеграцію з зовнішніми сервісами та мобільними клієнтами, а також дозволяє створювати повноцінні сторонні інтеграції.

Всі інструментальні засоби, застосовані у проєкті, згруповано у вигляді таблиці технологічного стеку (табл. 1.3).

Таблиця 1.3

Технологічний стек проєкту.

| Компонент системи         | Технологія / Засіб          | Призначення                                    |
|---------------------------|-----------------------------|------------------------------------------------|
| Мова програмування        | Java                        | Розробка бекенду                               |
| Фреймворк                 | Spring Boot                 | Створення REST API, інтеграція з БД            |
| Безпека                   | Spring Security + JWT       | Аутентифікація, авторизація                    |
| Комунікація між сервісами | OpenFeign                   | Легка побудова HTTP-клієнтів для мікросервісів |
| ORM                       | Spring Data JPA + Hibernate | Робота з БД, об'єктно-реляційне відображення   |
| Історія змін              | Hibernate Envers            | Відстеження змін у сутностях                   |
| Мапінг DTO - Entity       | MapStruct                   | Генерація мапінгу                              |
| Зменшення шаблонного коду | Lombok                      | Генерація гетерів, сетерів, конструкторів тощо |
| База даних                | PostgreSQL                  | Основне сховище даних                          |
| Міграції БД               | Flyway                      | Керування версіями структури БД                |
| Хмарне сховище            | Google Cloud Storage        | Збереження чеків, звітів, медіафайлів          |
| Інструменти для CI/CD     | GitHub Actions / Docker     | Автоматизація збірки, тестування, розгортання  |

У процесі розробки проєкту було використано сучасні інструменти розробника, що сприяють ефективності та якості коду. Основним середовищем розробки є IntelliJ IDEA з підтримкою Spring Boot, інтеграцією з Git, Docker та плагінами для перевірки коду. Для тестування REST API застосовувалися Postman і Insomnia, що дозволяє зручно перевіряти роботу

Вимоги до апаратного забезпечення.

Визначемо необхідну конфігурацію апаратного забезпечення для повноцінної розробки та локального розгортання проєкту (табл. 1.4).

Таблиця 1.4

| Компонент          | Мінімальні вимоги                                                                            | Рекомендовані вимоги                                            |
|--------------------|----------------------------------------------------------------------------------------------|-----------------------------------------------------------------|
| Процесор           | 4-ядерний                                                                                    | Intel Core i5 / AMD Ryzen 5                                     |
| Оперативна пам'ять | 8 ГБ                                                                                         | 16 ГБ для комфортної роботи з Docker, IDE та кількома сервісами |
| Накопичувач        | SSD 100 ГБ                                                                                   | SSD 200+ ГБ (з урахуванням залежностей, журналів, БД)           |
| Операційна система | Windows 10/11, macOS 12+, Ubuntu 22.04 LTS або новіше                                        | -                                                               |
| Інтернет-з'єднання | 10 Мбіт/с, стабільне для роботи з хмарними сервісами (Google Cloud, GitHub, Docker Hub тощо) | -                                                               |

Таким чином, обраний набір технологій забезпечує надійність, зручність розробки та підтримку сучасних вимог до розподілених веб-систем. А апаратні ресурси, що вимагаються, відповідають типовому рівню сучасного робочого середовища для розробника програмного забезпечення.

### Висновки до першого розділу.

У результаті аналізу предметної області визначено основні вимоги до CRM-системи для приватних шкіл, зосередженої на автоматизації фінансових процесів - управлінні платежами, звітністю та інтеграції з банківськими сервісами. Досліджено аналогічні рішення, проведено порівняння підходів до обробки транзакцій і фінансової аналітики. На основі отриманих даних сформовано перелік функціональних і нефункціональних вимог, а також обґрунтовано вибір технологій для створення надійної, масштабованої та зручної у підтримці системи.

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         | 22   |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         |      |

## РОЗДІЛ 2. ПРОЕКТУВАННЯ CRM-СИСТЕМИ ДЛЯ ОПТИМІЗАЦІЇ ФІНАНСОВИХ ПРОЦЕСІВ У ШКОЛАХ

### 2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи.

CRM-система розробляється з метою автоматизації ключових фінансових процесів: обліку платежів за навчання, генерації звітів, синхронізації з банківськими сервісами, а також зручного управління взаєморозрахунками з батьками та учнями.

Визначемо список ролей доступних в системі (табл. 2.1).

Таблиця 2.1

| Роль                | Опис                                                                                 |
|---------------------|--------------------------------------------------------------------------------------|
| Адміністратори      | Мають повний доступ до системи                                                       |
| Менеджери шкіл      | Створення платежів, перевірка оплат, контроль ставок зарплат, взаємодія з платниками |
| Батьки або платники | Перегляд рахунків, історії оплат, оплата навчання, отримання квитанцій               |
| Викладачі           | Доступ до розкладу, облік занять, перегляд зарплати та статистики                    |
| Офісні працівники   | Формування звітів, перегляд оплат, груп та платників                                 |

Для наочної демонстрації взаємодії між користувачами та системою розроблено діаграму варіантів використання (рис. 2.1), яка відображає основні сценарії взаємодії користувачів з CRM-системою. Ця діаграма дає змогу краще зрозуміти логіку роботи кожної ролі та зв'язки між функціональними модулями. Такий підхід дозволяє краще спроектувати логіку застосунку, забезпечити коректну розмежованість прав доступу, а також уникнути дублювання функціональності між модулями.

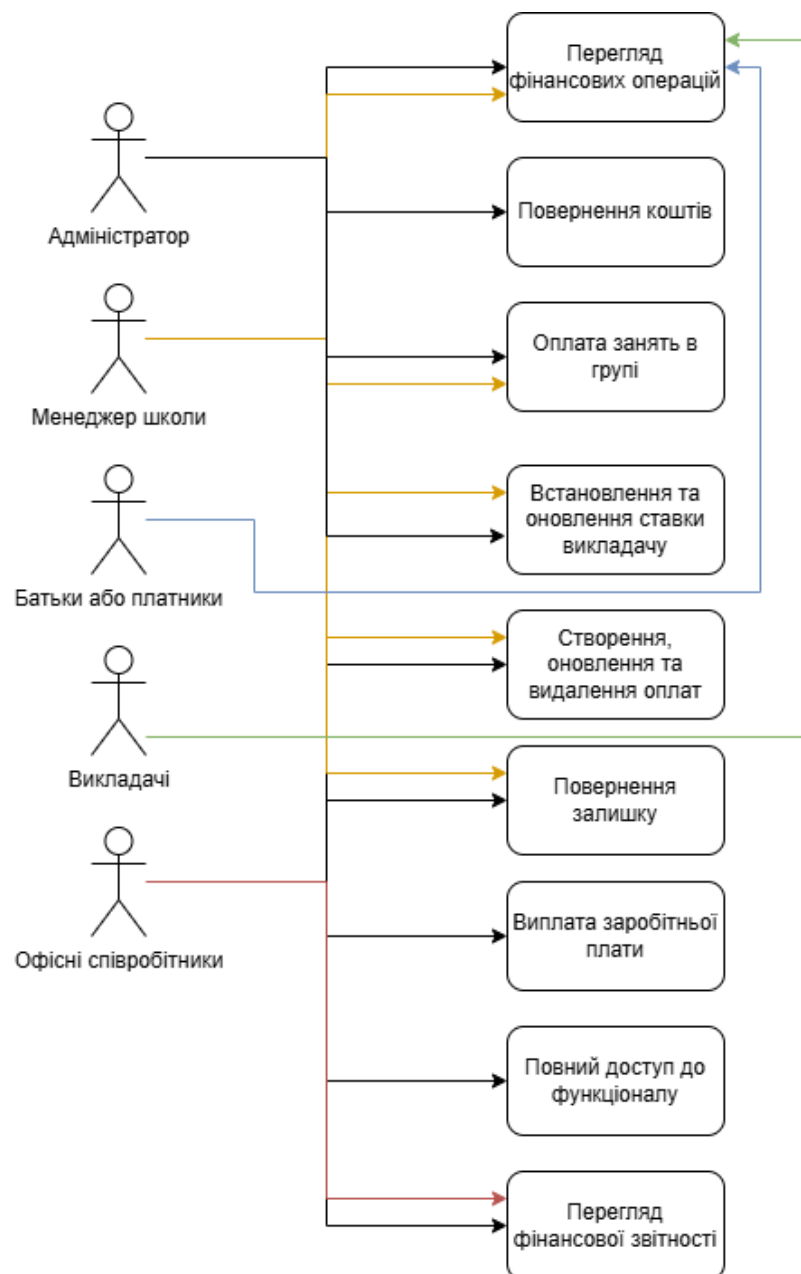


Рис. 2.1. Діаграма варіантів використання.

Опис діаграми варіантів використання наведено в таблицях 2.2 – 2.10.

Таблиця 2.2.

Опис прецеденту «Перегляд фінансових операцій»

|                             |                                      |
|-----------------------------|--------------------------------------|
| Назва варіанту використання | Перегляд фінансових операцій         |
| Основні актори              | Менеджер школи, учень, адміністратор |

## Продовження таблиці 2.2.

|               |                                                                                                                                               |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| Короткий опис | Користувачі можуть переглядати історію оплат, повернень, рознесення коштів по групах та інші фінансові дії, що стосуються їхніх прав доступу. |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------|

## Таблиця 2.3.

## Опис прецеденту «Створення, видалення, редагування оплат»

|                             |                                                                                                   |
|-----------------------------|---------------------------------------------------------------------------------------------------|
| Назва варіанту використання | Створення, видалення, редагування оплат                                                           |
| Основні актори              | Менеджер школи, Адміністратор                                                                     |
| Короткий опис               | Дозволяє створювати нові оплати, редагувати помилкові платежі або видаляти некоректні транзакції. |

## Таблиця 2.4.

## Опис прецеденту «Повернення залишку»

|                             |                                                                                               |
|-----------------------------|-----------------------------------------------------------------------------------------------|
| Назва варіанту використання | Повернення залишку                                                                            |
| Основні актори              | Менеджер школи, Адміністратор                                                                 |
| Короткий опис               | Дає змогу повернути частину коштів користувачу, якщо є переплата або інші залишки на рахунку. |

## Таблиця 2.5.

## Опис прецеденту «Повернення коштів»

|                             |                               |
|-----------------------------|-------------------------------|
| Назва варіанту використання | Повернення коштів             |
| Основні актори              | Менеджер школи, Адміністратор |

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         | 25   |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         |      |

## Продовження таблиці 2.5.

|               |                                                                                                     |
|---------------|-----------------------------------------------------------------------------------------------------|
| Короткий опис | Повне повернення оплати у випадках, коли послуга не була надана або за індивідуальними обставинами. |
|---------------|-----------------------------------------------------------------------------------------------------|

## Таблиця 2.6.

## Опис прецеденту «Рознесення коштів по групах»

|                             |                                                                  |
|-----------------------------|------------------------------------------------------------------|
| Назва варіанту використання | Рознесення коштів по групах                                      |
| Основні актори              | Менеджер школи, Адміністратор                                    |
| Короткий опис               | Автоматичне або ручне розподілення коштів за навчальними групами |

## Таблиця 2.7.

## Опис прецеденту «Встановлення та оновлення ставки викладачу»

|                             |                                                                                          |
|-----------------------------|------------------------------------------------------------------------------------------|
| Назва варіанту використання | Встановлення та оновлення ставки викладачу                                               |
| Основні актори              | Менеджер школи, Адміністратор                                                            |
| Короткий опис               | Дозволяє призначати та змінювати погодинну ставку або фіксовану оплату праці викладачам. |

## Таблиця 2.8.

## Опис прецеденту «Виплата заробітної плати викладачу»

|                             |                                                                 |
|-----------------------------|-----------------------------------------------------------------|
| Назва варіанту використання | Виплата заробітної плати викладачу                              |
| Основні актори              | Адміністратор                                                   |
| Короткий опис               | Автоматизований розрахунок та фіксація виплати заробітної плати |

Таблиця 2.9.

## Опис прецеденту «Перегляд фінансових звітів»

|                             |                                                                                                                 |
|-----------------------------|-----------------------------------------------------------------------------------------------------------------|
| Назва варіанту використання | Перегляд фінансових звітів                                                                                      |
| Основні актори              | Офісний співробітник, Адміністратор                                                                             |
| Короткий опис               | Дозволяє формувати, переглядати та експортувати звіти про надходження, витрати, оплату праці та інші показники. |

Таблиця 2.10.

## Опис прецеденту «Повний доступ до всього функціоналу»

|                             |                                                                                                                                        |
|-----------------------------|----------------------------------------------------------------------------------------------------------------------------------------|
| Назва варіанту використання | Повний доступ до всього функціоналу                                                                                                    |
| Основні актори              | Адміністратор                                                                                                                          |
| Короткий опис               | Надає адміністратору повні права для керування всіма аспектами системи — включно з користувачами, платежами, звітами та конфігурацією. |

Кожен із описаних варіантів використання у таблицях 2.2–2.10 виконує не лише окрему функцію в системі, а й відіграє роль складової частини ширшого бізнес-процесу у фінансовому управлінні навчальним закладом. У сукупності ці дії формують цілісну логіку обробки фінансових операцій - від створення рахунку до генерації звіту та виплати заробітної плати.

Головна ціль розробки CRM-системи для приватних освітніх закладів полягає у створенні цифрового середовища, яке дозволяє автоматизувати та оптимізувати фінансові процеси, пов'язані з навчанням, платежами та звітністю. Система покликана зменшити навантаження на адміністративний персонал, підвищити точність обробки платежів та забезпечити прозорість усіх фінансових операцій у школі.

|      |      |                   |        |      |                                         |      |
|------|------|-------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А. |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.    |        |      |                                         |      |
| Змн. | Арк. | № докум.          | Підпис | Дата |                                         | 27   |

Запропонована платформа дає змогу менеджерам шкіл зручно керувати платежами, здійснювати повернення коштів, розподіляти платежі між навчальними групами та фіксувати залишки. Крім цього, працівники, зокрема бухгалтери, мають інструменти для формування та перегляду звітів, що дозволяє ефективно контролювати фінансовий стан філій.

Ключовим завданням розробки є забезпечення централізованого управління фінансовими потоками навчального закладу та надання різним категоріям користувачів (менеджерам, адміністраторам, бухгалтерам, учням) зручного доступу до актуальної інформації. Реалізація такої системи сприятиме підвищенню рівня довіри до закладу з боку батьків та учнів, покращить внутрішні бізнес-процеси і дозволить адаптуватися до сучасних цифрових вимог в освіті.

## 2.2. Розробка бази даних.

Етап проєктування та реалізації бази даних є фундаментальним у створенні CRM-системи, оскільки саме база даних забезпечує централізоване зберігання, обробку та доступ до всієї фінансової, адміністративної та навчальної інформації. Враховуючи специфіку системи, яка покликана автоматизувати фінансові процеси у школах - такі як облік оплат, формування звітів, взаємодія з зовнішніми API, тощо - важливість правильно побудованої БД є надзвичайно високою.

У процесі проєктування та вибору технологій для побудови CRM-системи було проаналізовано кілька популярних систем управління базами даних (СУБД). Основна мета — забезпечити стабільність, масштабованість, безпеку та підтримку складних транзакцій і зв'язків. Порівняємо найпопулярніші СУБД (табл. 2.11).

Таблиця 2.11.

| Критерій | PostgreSQL         | MariaDB   | MySQL     | MongoDB (NoSQL)        |
|----------|--------------------|-----------|-----------|------------------------|
| Тип СУБД | Об'єктно-реляційна | Реляційна | Реляційна | Документно-орієнтована |

Продовження таблиці 2.11.

| Підтримка транзакцій       | Повна ACID-підтримка                                                   | Повна (залежно від рушія, напр. InnoDB)       | Повна (з InnoDB)                                   | Обмежена, не ACID за замовчуванням                         |
|----------------------------|------------------------------------------------------------------------|-----------------------------------------------|----------------------------------------------------|------------------------------------------------------------|
| Масштабованість            | Висока, завдяки підтримці реплікацій, партицій та розподілених запитів | Висока, з хорошою підтримкою реплікації       | Висока, але масштабування горизонтальне складніше  | Висока, природно горизонтально масштабована                |
| Підтримка складних запитів | Дуже добра: CTE, віконні функції, JSONB, stored procedures             | Добра, але менш потужна, ніж у PostgreSQL     | Добра, але дещо поступається PostgreSQL            | Обмежена: запити гнучкі, але не реляційні                  |
| JSON-поля та гнучкість     | Повна підтримка JSONB з індексацією                                    | Часткова підтримка JSON                       | Є підтримка JSON, але без індексації               | Природна підтримка, основний формат зберігання             |
| Механізми безпеки          | Потужні засоби контролю доступу, ролі, SSL                             | Стандартні механізми                          | Стандартні механізми                               | Шифрування, ролі, але потребує додаткових налаштувань      |
| Переваги                   | Надійність, підтримка складних операцій, чудова інтеграція з Java      | Швидша у деяких операціях, сумісність з MySQL | Популярність, активна спільнота                    | Добра для гнучкої, нереляційної структури, масштабованість |
| Недоліки                   | Більш складна для освоєння                                             | Обмеження при роботі з JSON, менше функцій    | Менше можливостей для складних аналітичних запитів | Не підходить для складних реляційних моделей               |

| Ліцензія | Open Source<br>(PostgreSQL License) | Open Source<br>(GPL v2) | Open Source<br>(GPL v2, але під<br>Oracle) | Open Source (SSPL, із<br>комерційними<br>обмеженнями) |
|----------|-------------------------------------|-------------------------|--------------------------------------------|-------------------------------------------------------|
|----------|-------------------------------------|-------------------------|--------------------------------------------|-------------------------------------------------------|

Для реалізації бази даних було обрано PostgreSQL як одну з найбільш надійних, продуктивних та відкритих об'єктно-реляційних СУБД. Вона надає широкі можливості роботи зі складними запитам, дозволяє використовувати транзакції, тригери, зберігає процедури та має потужну підтримку JSONB-структур, що зручно при зберіганні логів і параметрів.

База даних дозволяє впорядкувати всі ключові об'єкти предметної області: інформацію про учнів, навчальні групи, курси, оплати, повернення, залишки, фінансову звітність та ставки викладачів. Це дає змогу користувачам різних ролей швидко отримувати доступ до актуальної, цілісної та взаємопов'язаної інформації, що є критично важливим для прийняття управлінських рішень.

Оскільки система реалізована за принципами мікросервісної архітектури, кожен сервіс (наприклад, сервіс обробки платежів, обліку ставок чи генерації звітів) працює зі своєю логічно відокремленою частиною даних. Це дозволяє досягти високого рівня ізоляції модулів, спрощує масштабування, забезпечує локалізацію помилок, а також покращує продуктивність, адже кожен мікросервіс взаємодіє лише з релевантною частиною схеми.

Завдяки обраному механізму проєктування та використанню PostgreSQL, система має змогу витримувати зростання навантаження при збільшенні кількості учнів, платежів та навчальних груп без втрати швидкодії. Ретельно спроектовані індекси, зв'язки між таблицями та оптимізовані SQL-запити забезпечують високу ефективність при обробці великих обсягів даних.

На цьому етапі здійснюється збір функціональних та нефункціональних вимог: які ролі працюватимуть із системою, які типи даних потрібно, яка частота оновлення даних, які бізнес-процеси мають бути підтримані.

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         | 30   |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         |      |

Було розроблено логічну та фізичну модель бази даних із чітко визначеними таблицями, атрибутами, типами даних, зовнішніми ключами та індексами. Основні таблиці включають: users, students, payments, refunds, salary, transactions, reports, groups, teachers, tariffs, logs. Сутності поєднані у відповідності до предметної області, з урахуванням нормалізації до 3NF.

Усі зміни схеми БД керуються через систему Flyway, що дозволяє поступово розгортати, оновлювати або відновлювати БД, зберігаючи контроль над усіма версіями. SQL-скрипти впорядковано та перевірено на кожному етапі оновлення застосунку.

Завдяки інтеграції Hibernate Envers реалізовано повноцінний аудит змін: кожна зміна у фінансових транзакціях, оплатах або звітах фіксується в окремій таблиці історії. Це забезпечує прозорість і можливість відновлення даних при помилках чи порушеннях.

Після реалізації початкової версії схеми проведено тестування продуктивності на даних, що імітують реальне навантаження. В результаті були виявлені та усунені "вузькі місця", оптимізовані запити та додано індексацію найбільш використовуваних полів.

База даних розгортається за допомогою Docker-контейнерів або в хмарному середовищі (наприклад, Google Cloud SQL), а доступ до неї організовано через REST API, побудовані на Spring Boot з використанням OpenFeign

Особливу роль відіграє бібліотека MapStruct - вона автоматизує процес перетворення між DTO-об'єктами, які отримує фронтенд, і сутностями, що зберігаються в базі. Наприклад, StudentDto може бути автоматично перетворений у сутність Student, яка потім зберігається через Spring Data JPA у таблиці students. Такий підхід знижує кількість помилок, покращує читаність коду та спрощує підтримку.

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         | 31   |

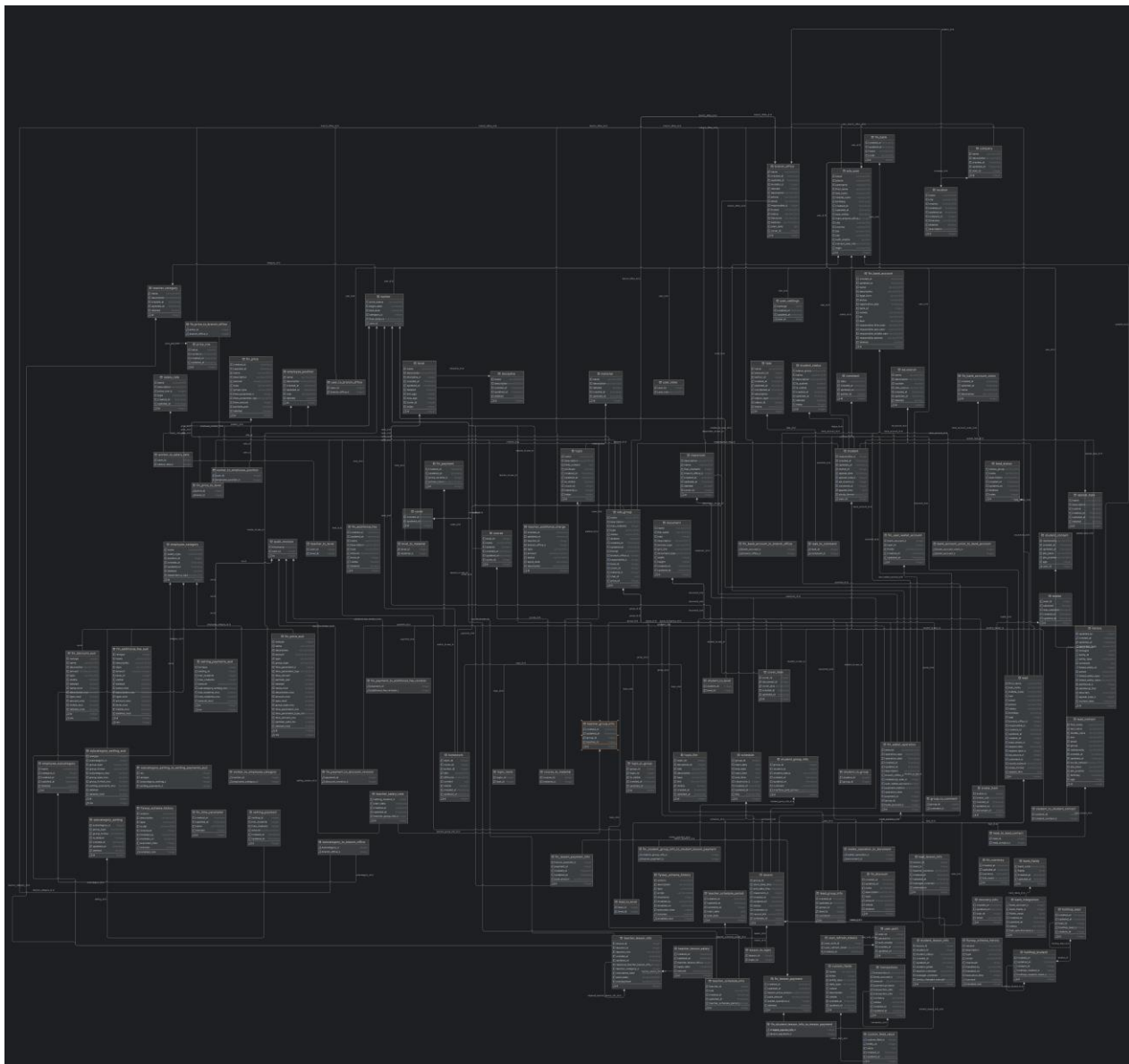


Рис.2.2. Діаграма бази даних.

Таблиця. 2.12.

Опис таблиць бази даних.

| Назва таблиці | Опис                                                                     |
|---------------|--------------------------------------------------------------------------|
| report        | Зберігає різні звіти щодо діяльності системи (фінансові, навчальні).     |
| transactions  | Зберігає фінансові транзакції в системі, включно зі статусом транзакції. |
| fin_currency  | Довідник валют, що використовуються у фінансових операціях.              |

## Продовження таблиці 2.12.

|                              |                                                                      |
|------------------------------|----------------------------------------------------------------------|
| fin_bank_account             | Банківські рахунки.                                                  |
| fin_lesson_payment           | Дані про оплату занять студентом.                                    |
| setting_payment              | Налаштування оплат викладачів для автоматичного встановлення ставки. |
| fin_additional_fee           | Додаткові платежі: штрафи, знижки, послуги.                          |
| salary_rate                  | Ставка заробітної плати для викладачів.                              |
| fin_lesson_payment_info      | Детальна інформація про оплату занять студентом.                     |
| fin_user_wallet_account      | Гаманець користувача: кількість коштів та пов'язані операції.        |
| fin_bank                     | Перелік банків, що використовуються в системі.                       |
| employee_subcategory         | Підкатегорії працівників (великі, середні, малі міста).              |
| teacher_additional_charge    | Додаткові нарахування викладачам.                                    |
| fin_discount                 | Знижки для клієнтів.                                                 |
| wallet_operation_to_document | Зв'язок фінансових операцій із документами.                          |
| notification                 | Сповіщення для користувачів.                                         |
| subcategory_setting          | Налаштування для підкатегорій.                                       |
| fin_payment                  | Загальні записи про платежі.                                         |
| teacher_lesson_salary        | Нарахування заробітної плати за уроки.                               |
| fin_price                    | Ціни для груп.                                                       |
| bank_integration             | Дані для інтеграції з банківськими сервісами.                        |
| bank_fields                  | Поля для інтеграції з банківськими системами.                        |
| fin_time_parameter           | Параметри часу для фінансових розрахунків.                           |
| history                      | Дані змін різних сутностей у системі.                                |
| fin_wallet_operation         | Внутрішні операції з гаманцем користувача.                           |

У процесі створення структури бази даних було враховано не лише вимоги предметної області, а й загальні принципи побудови корпоративних інформаційних систем. Зокрема, велика увага приділялася дотриманню нормалізації даних, що дозволяє уникати дублювання та підвищує цілісність інформації. Була передбачена оптимізація запитів через створення індексів на часто використовувані поля, використання віконних функцій, матеріалізованих представлень та переглядів для прискорення агрегованих операцій. Усі сутності були логічно згруповані відповідно до їх призначення - фінансові об'єкти, навчальні структури, користувачі. Це дозволяє масштабувати систему гнучко, додаючи нові модулі без порушення цілісності.

Для забезпечення цілісності даних активно використовувалися обмеження на рівні СУБД, такі як зовнішні ключі, унікальні поля, перевірки типу CHECK, а також NOT NULL для критичних атрибутів. Це дало змогу зменшити ймовірність виникнення логічних помилок або непослідовностей. Окремі процеси автоматизовано за допомогою тригерів — наприклад, автоматичне оновлення балансу користувача після надходження оплати, або зміна статусу фінансової транзакції після підтвердження від зовнішнього банку. Такі рішення не лише спрощують логіку бекенду, а й гарантують синхронізацію змін на рівні самої БД.

Враховуючи те, що архітектура системи побудована за мікросервісним принципом, кожен сервіс взаємодіє лише з логічно виділеною частиною даних. Це дозволяє уникати конфліктів при масштабуванні, спрощує технічне обслуговування. Дані між сервісами передаються через API, а не прямим доступом до спільної бази - такий підхід забезпечує слабке зв'язування та підвищену безпеку.

Особливу роль у реалізації бази даних відіграє система версіонування Flyway. Кожна зміна структури зберігається у вигляді окремого SQL-скрипту з відповідним номером версії, що дозволяє точно відстежувати, які зміни коли були впроваджені. Це особливо важливо при роботі над системою командою.

Таким чином, база даних у даній CRM-системі - це не просто сховище інформації, а повноцінний механізм, який реалізує бізнес-логіку, забезпечує

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         | 34   |

прозорість, контроль, масштабованість і аналітику. Вона є фундаментом, що дозволяє адаптувати систему під специфіку роботи різних шкіл та забезпечує ефективну автоматизацію фінансових процесів.

### 2.3 Реалізація системи.

На цьому етапі кваліфікаційної роботи розпочинається безпосередня реалізація програмного забезпечення. Усі організаційні та технічні аспекти, пов'язані з розробкою CRM-системи для приватних шкіл, вже були опрацьовані на попередніх етапах. Сформульовано чіткий перелік функціональних завдань, основна мета яких - автоматизація та оптимізація фінансових процесів. До таких завдань належать: облік оплат, нарахування знижок, синхронізація з банківськими сервісами, повернення коштів, формування звітності, управління залишками тощо. Також було визначено загальний технічний стек, підхід до архітектури системи, та способи реалізації кожного функціонального модуля. Відтак, з урахуванням поставлених цілей і вимог, можна переходити до безпосередньої реалізації системи.

На початку розробки необхідно створити структуру бази даних. Для цього використовується інструмент Flyway, який забезпечує контрольовану міграцію бази даних. Усі міграції створюються в SQL-форматі та додаються в репозиторій проекту, щоб забезпечити повну повторюваність на будь-якому середовищі. У базі даних створюються таблиці, які зберігатимуть інформацію про користувачів, студентів, банківські рахунки, гаманці, фінансові операції, групи, тарифи, історії змін, транзакції тощо. Важливо, що вся фінансова активність фіксується у відповідних історіях, які дозволяють не тільки здійснювати аудит, а й відновлювати дані в разі необхідності. Для зручності масштабування і спрощення роботи з оточенням, усі сервіси та БД запускаються в Docker-контейнерах.

Для забезпечення взаємодії між окремими мікросервісами, використовується технологія Feign - це зручний інструмент, який дозволяє організовувати виклики HTTP-запитів у вигляді простих Java-інтерфейсів. Замість

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         | 35   |

того щоб вручну створювати REST-клієнтів, формувати запити та обробляти відповіді, ми просто описуємо потрібний метод у вигляді звичайної Java-функції. Це значно спрощує код, покращує його читабельність та підтримку, а також знижує ризик помилок. Наприклад, коли CRM-сервісу потрібно надіслати запит до сервісу інтеграції, щоб дізнатися, чи успішно пройшла синхронізація для певного банківського рахунку, він просто викликає відповідний метод `integrationClient.getSyncStatus(accountId)`, де `integrationClient` - це інтерфейс `Feign`-клієнта з відповідною анотацією `@FeignClient`.

Кожен такий інтерфейс містить чітко визначені маршрути, HTTP-методи і параметри, тому будь-яка взаємодія між сервісами відбувається прозоро та уніфіковано. Крім того, у разі розширення логіки, нові методи легко додаються до клієнта без потреби переписувати всю взаємодію з нуля.

Завдяки `Feign` обмін даними між мікросервісами залишається слабо зв'язаним, тобто сервіси можуть незалежно масштабуватися або розгортатися, не порушуючи загальну роботу системи. Це цілком відповідає принципам мікросервісної архітектури, яку було обрано при проектуванні системи.

Для обробки виключень створено набір кастомних винятків, які наслідують базовий клас `RuntimeException`. Кожен виняток асоційовано з унікальним кодом помилки, який дозволяє швидко ідентифікувати тип проблеми та місце її виникнення. Усі виключення обробляються за допомогою глобального хендлера `@ControllerAdvice`, який перехоплює всі не перехоплені помилки на рівні контролера. У відповідь клієнту повертається структурований об'єкт помилки (`ErrorResponse`), що містить:

- HTTP-статус (наприклад, 400, 404, 409)
- Унікальний код помилки
- Повідомлення помилки
- Додаткову технічну інформацію

Це забезпечує єдину точку входу для обробки всіх помилок у системі та дозволяє розробникам легко відстежувати, логувати й відлагоджувати виклики.

|      |      |                   |        |      |                                         |      |
|------|------|-------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А. |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.    |        |      |                                         |      |
| Змн. | Арк. | № докум.          | Підпис | Дата |                                         | 36   |

Логування реалізовано за допомогою бібліотеки SLF4J, що дозволяє гнучко налаштовувати рівні логування залежно від середовища. Усі ключові події, включаючи успішні та неуспішні запити, зберігаються в логах із вказанням часу, користувача, шляху запиту, параметрів, статусу відповіді тощо.

Ще одним важливим аспектом є конфігурація поведінки сервісів залежно від середовища розгортання. У CRM-системі реалізовано профільну конфігурацію, що дає змогу легко адаптувати роботу застосунку до різних середовищ: локального, тестового, розробницького та продакшн.

Кожен мікросервіс має кілька конфігураційних файлів типу:

- application.yml — для продакшена.
- application-local.yml — для локального середовища розробки.
- application-dev.yml — для середовища розробників.

Усі файли містять конфігурацію, специфічну для відповідного середовища.

Подальша реалізація системи базується на підході багаторівневої архітектури. Зокрема, створюються DTO-класи для обміну даними між клієнтом і сервером, моделі для опису структури таблиць у Java, мапери - для перетворення DTO в Entity і навпаки, контролери - для обробки HTTP-запитів, репозиторії - для доступу до БД, а також сервіси, в яких реалізується основна бізнес-логіка.

Репозиторії реалізуються через Spring Data JPA, з використанням інтерфейсів, які розширюють JpaRepository. Оскільки стандартні CRUD-операції надаються автоматично, більшість репозиторіїв не містить додаткової логіки, за винятком специфічних запитів. Контролери, мапери та утиліти реалізуються відповідно до принципів простоти та розділення відповідальностей. Основна бізнес-логіка зосереджена в сервісах, де відбувається обробка всіх фінансових сценаріїв системи.

Деякі сервіси, такі як BankService, BankAccountService та BankAccountUnionService, не містять складної логіки — вони реалізують лише стандартні CRUD-операції: створення, оновлення, видалення та отримання інформації про банківські рахунки, прив'язки рахунків і самі банки. Ці сервіси

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         | 37   |

виконують допоміжну роль, забезпечуючи базову функціональність для роботи з фінансовими даними.

Одним з ключових компонентів системи є сервіс WalletOperationService, який відповідає за обробку фінансових операцій користувачів. Цей сервіс опрацьовує зарахування коштів на особистий гаманець користувача. Якщо гаманець ще не створений — він створюється автоматично. Після зарахування створюється запис в історії операцій, у якому фіксуються тип операції, сума, час, користувач та коментар. Також сервіс відповідає за повернення коштів користувачу. Перед тим, як виконати операцію, проводиться перевірка наявності достатньої суми в гаманці. У разі успішної перевірки кошти списуються з балансу, і створюється відповідний запис в історії.

Більш складний сценарій реалізований у процесі оплати занять. Якщо під час цієї операції користувач обирає оплату новим платежем, то спочатку виконується інша операція, а саме зарахування коштів на особистий рахунок. Після чого знову запускається операція списання коштів на заняття, але вже з доступними коштами. Після цього система автоматично відфільтровує усі неоплачені заняття, які відповідають обраним параметрам, і по кожному з них виконується розрахунок вартості. При цьому враховуються індивідуальні знижки, тип оплати (разова чи абонемент), а також можливі доплати. Для кожного заняття перевіряється, чи вистачає коштів на повну оплату. Якщо група не підтримує часткову оплату, а суми не вистачає — заняття пропускається. Якщо часткова оплата дозволена або коштів достатньо, заняття оплачується, і відповідна сума списується з загальної суми платежу. Таким чином, автоматично покриваються всі заняття, на які вистачає коштів. При цьому обов'язково перевіряється, чи взагалі потрібно оплачувати конкретне заняття, адже в системі передбачено і безкоштовні заняття, які не потребують оплати (наприклад, пробні чи подарункові). Після завершення оплати кожному заняттю присвоюється відповідний статус, а інформація про дату, час, суму, тип платежу та інші деталі зберігається в базі даних.

|      |      |                   |        |      |                                         |      |
|------|------|-------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А. |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.    |        |      |                                         |      |
| Змн. | Арк. | № докум.          | Підпис | Дата |                                         | 38   |

Також якщо в результаті оплати занять залишаються кошти, які не можуть бути розподілені через відсутність занять, вони формують залишок. Цей залишок можна буде повернути окремо за допомогою відповідної операції. Після кожної з описаних операцій фіксується запис в історії як у студента, так і у групи. Всі операції поділені на окремі методи, що дозволяє спростити підтримку та перевикористання коду відповідно до принципів SOLID - зокрема принципу єдиної відповідальності.

Окремо слід зупинитися на процесі синхронізації оплат із банківською системою. Цей функціонал реалізовано у вигляді окремого мікросервісу під назвою *integration*. Він відповідає за автоматичне отримання банківських транзакцій через API та прив'язку їх до студентів у CRM-системі. Щоб розпочати синхронізацію, потрібно мати створений банківський рахунок у системі з налаштуваннями доступу до API, а також успішно пройдену перевірку інтеграції (тобто статус рахунку має бути “success”). Після виконання цих умов можна запускати процес синхронізації.

Оскільки обробка банківських транзакцій може тривати довго, залежно від кількості нових записів - вона реалізована в асинхронному режимі. Крім того, впроваджено захист від одночасного запуску синхронізації за допомогою *AtomicBoolean*. Це дозволяє уникнути дублювання процесів. Під час синхронізації кожна транзакція витягується з API банку, обробляється, і з її опису за допомогою регулярних виразів витягується ID студента. Якщо студент знайдений — транзакція прикріплюється до його гаманця, і йому автоматично зараховуються кошти. Якщо знайти студента не вдається — транзакція отримує статус “Потребує уваги”, і адміністратор може пізніше вручну вказати потрібного студента, після чого система самостійно виконає зарахування.

Після завершення синхронізації оновлюється її статус і фіксується дата останнього оновлення. Це дозволяє адміністраторам бачити успішність останньої інтеграції та мати орієнтовну дату наступної. Таким чином, функціонал інтеграції забезпечує повну автоматизацію надходження оплат у систему без участі людини, з можливістю ручної корекції лише в окремих виняткових випадках. Це суттєво

|      |      |                   |        |      |                                         |      |
|------|------|-------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А. |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.    |        |      |                                         |      |
| Змн. | Арк. | № докум.          | Підпис | Дата |                                         | 39   |

знижує навантаження на бухгалтерію та мінімізує ймовірність помилок при внесенні платежів вручну.

Крім фінансових операцій, у системі реалізовано механізм формування звітності, який допомагає адміністраторам і фінансовому відділу контролювати всі грошові потоки та виконані нарахування. За це відповідає окремий сервіс під назвою ReportService. У поточній версії системи реалізовано три основні звіти.

Таблиця 2.13

| Назва звіту                | Опис                                                                                       | Основні дані                                                                  | Доступні фільтри                                   |
|----------------------------|--------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------|----------------------------------------------------|
| Звіт про оплачений час     | Показує, скільки занять було проведено й оплачено, скільки часу пройдено по кожному тарифу | - Учень<br>- Група<br>- Кількість занять<br>- Оплачений час                   | - Період<br>- Філія<br>- Група<br>- Учень          |
| Звіт про клієнтські оплати | Відображає усі платежі, що надійшли від клієнтів                                           | - Клієнт<br>- Дата оплати<br>- Сума<br>- Спосіб оплати (готівка, безготівка)  | - Період<br>- Філія<br>- Спосіб оплати<br>- Клієнт |
| Звіт про заробітні плати   | Відображає зарплату викладачам на основі проведених занять, ставок та бонусів              | - Викладач<br>- Кількість занять<br>- Ставка<br>- Сума до виплати<br>- Бонуси | - Період<br>- Філія<br>- Група<br>- Викладач       |

### Висновки до другого розділу.

У цьому розділі було проведено розробку загальної архітектури та функціональних алгоритмів CRM-системи для приватних шкіл. Детально описано основні модулі. Розроблено структуру бази даних, реалізовано автоматизовану обробку платежів, знижок, нарахування зарплат та генерацію звітності.

У результаті виконаної роботи отримано повноцінний програмний продукт, який відповідає поставленим завданням та дозволяє ефективно автоматизувати ключові фінансові процеси у шкільній CRM-системі.

## РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З CRM-СИСТЕМОЮ ДЛЯ АВТОМАТИЗАЦІЇ ФІНАНСОВИХ ПРОЦЕСІВ У ШКОЛІ

### 3.1 Структура інтерфейсу системи та тестування системи.

Інтерфейс CRM-системи побудований з урахуванням принципів зручності, логічної структури та швидкого доступу до основних функцій. Завдяки адаптивному дизайну інтерфейс коректно відображається як на десктопних, так і на мобільних пристроях.

Розглядати алгоритм роботи системи будемо з ролі адміністратора, оскільки саме він має повний доступ до всіх функціональних можливостей, включаючи керування користувачами, налаштування, контроль доступу до модулів, запуск системних звітів та моніторинг фінансової діяльності. Це дозволяє оцінити всю функціональність CRM-системи та показати логіку взаємодії між модулями.

Розглянемо сторінку авторизації користувача в системі (рис. 3.1).

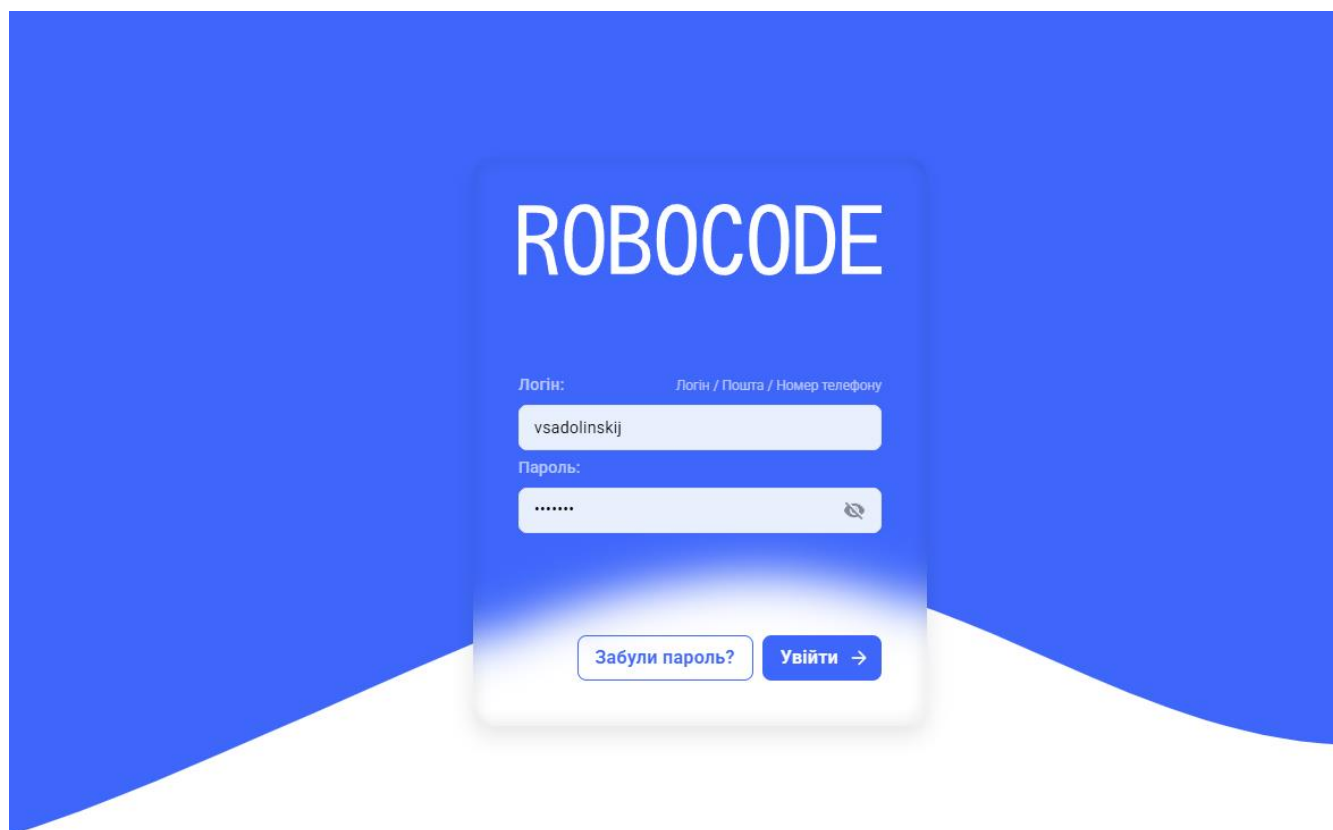


Рис. 3.1. Сторінка авторизації користувача.

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         | 41   |

На сторінці представлено форму для введення даних користувача, яка забезпечує різні способи входу до системи. Користувач може авторизуватися, використовуючи свій логін, адресу електронної пошти або номер телефону, що надає зручність та гнучкість при доступі до системи. Якщо користувач забув пароль, йому надається можливість його відновити. Для цього необхідно натиснути на кнопку «Забули пароль», яка розташована ліворуч від кнопки «Увійти».

Після натискання на цю кнопку користувач перенаправляється до форми відновлення паролю, яка відображається на сторінці (рис. 3.2). У цій формі потрібно вказати відповідні дані, що дозволять підтвердити особу та розпочати процес відновлення доступу до системи. Такий підхід забезпечує надійність і безпеку, а також мінімізує можливі проблеми користувачів із забутими обліковими даними, зберігаючи при цьому зручність взаємодії з системою.

Рис. 3.2. Форма відновлення пароля.

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         | 42   |

Після того, як користувач заповнить необхідні поля у формі та натисне кнопку «Відправити лист», система автоматично відправить на вказану електронну адресу листа з детальними інструкціями для відновлення доступу до облікового запису. У цьому листі буде міститися посилання, за допомогою якого користувач зможе безпечно та швидко скинути свій пароль і повернути доступ до системи.

Надсилання такого повідомлення гарантує, що лише власник вказаної електронної адреси може розпочати процес відновлення, що підвищує рівень безпеки та захисту персональних даних. Таким чином, користувач отримує чіткі вказівки для подальших дій, що дозволяє мінімізувати ризик втрати доступу до системи через забутий пароль і забезпечує безперервність роботи з платформою.

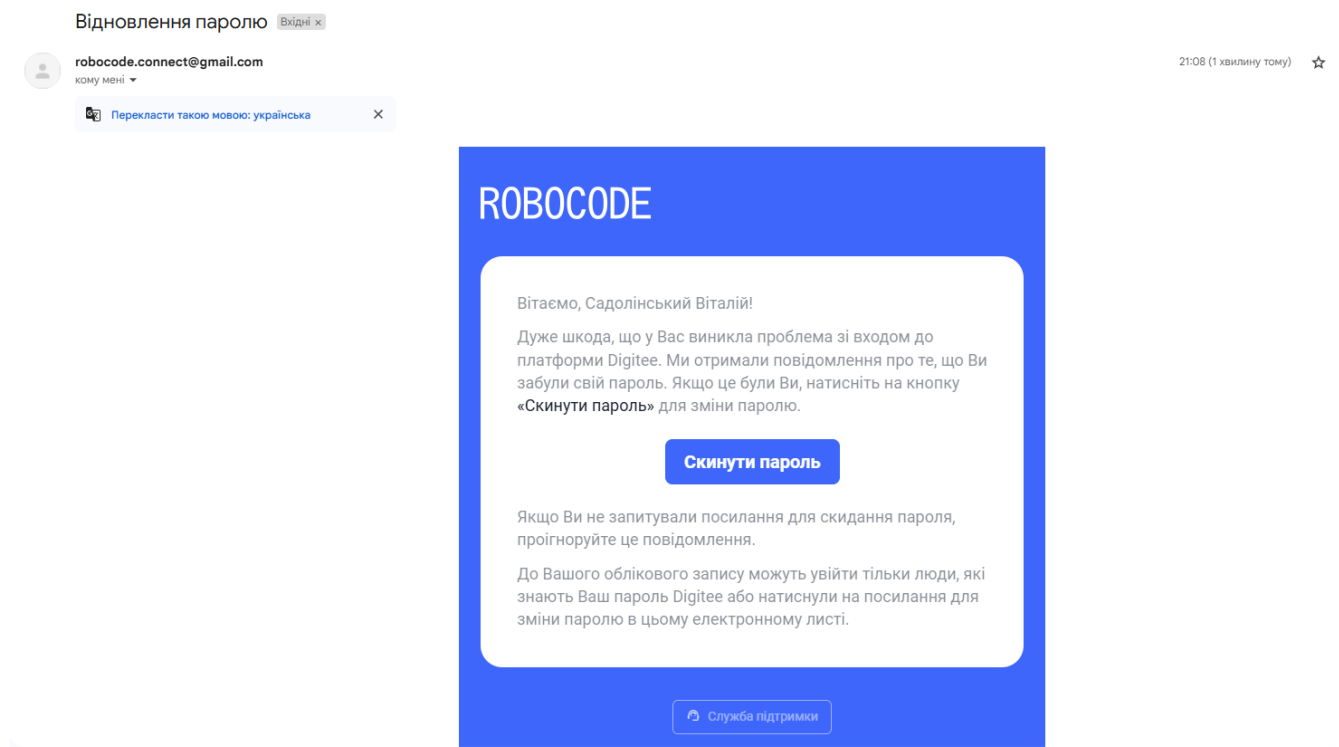


Рис. 3.3. Електронний лист для відновлення пароля.

Розглянемо головну сторінку, куди потрапляє користувач, після авторизації (рис. 3.4).

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолинський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         | 43   |

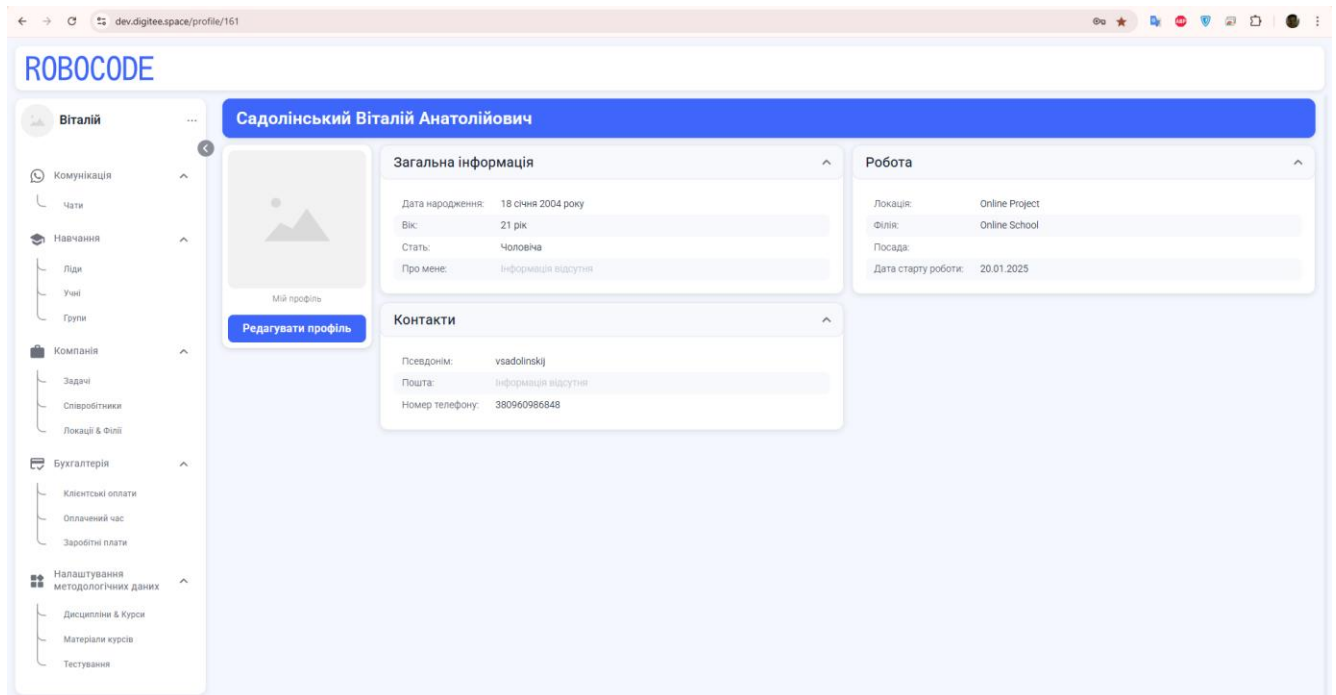


Рис. 3.4. Головна сторінка.

Головна сторінка системи включає в себе кілька ключових елементів, які забезпечують зручну навігацію та роботу користувача з додатком. Насамперед, тут присутня навігаційна панель, яка виконана у простому стилі, що складається лише з однієї кнопки з назвою компанії. Такий мінімалістичний підхід дозволяє не відволікати увагу користувача від основного контенту та спрощує інтерфейс. Окрім цього, на сторінці є бокове меню, яке містить основні пункти навігації, що допомагають користувачу швидко переходити між різними розділами системи. Робоча область займає центральне місце і слугує для відображення різноманітної інформації, у даному випадку - даних про користувача.

На цій сторінці користувач отримує доступ до ряду функцій, які підвищують зручність використання системи. Зокрема, він має можливість змінити свою роль у системі, скориставшись кнопкою у вигляді трьох крапок, розташованою у правому верхньому куті бокового меню (рис. 3.5). Це дозволяє швидко переключатися між різними рівнями доступу без необхідності додаткових дій. Також передбачено можливість переходу на будь-яку сторінку зі списку доступних, що робить навігацію по системі максимально гнучкою.

Важливою опцією є персоналізація робочої області, що дозволяє користувачеві налаштувати відображення блоків, змінюючи їх розташування та

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         | 44   |

розмір відповідно до власних вподобань і потреб. Це сприяє підвищенню продуктивності, оскільки користувач може організувати робочий простір найбільш комфортним для себе способом. Також в боковому меню є кнопка виходу, що дає змогу безпечно завершити роботу в системі. Загалом, головна сторінка створена таким чином, щоб забезпечити інтуїтивно зрозумілий та функціональний інтерфейс, який сприяє ефективній роботі користувача.

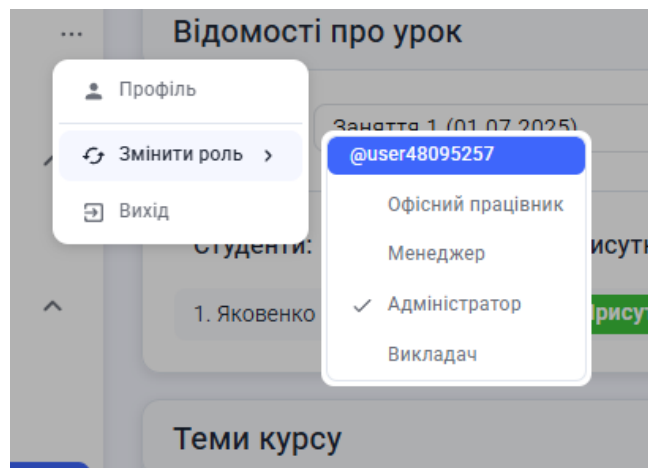


Рис. 3.5. Меню для зміни ролі користувача.

При натисненні на кнопку «Редагувати профіль» відкриється сторінка для редагування особистої інформації (рис. 3.6).

Рис. 3.6. Сторінка редагування особистої інформації.

Після того, як користувач натискає на кнопку «Редагувати профіль», відкривається окрема сторінка, призначена для внесення змін до особистої

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         | 45   |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         |      |

інформації. Ця сторінка надає зручний інтерфейс, який дозволяє користувачу оновити свої дані, такі як ім'я, прізвище, контактні відомості. Також можна змінити пошту та пароль (рис. 3.7).

Завдяки цьому користувач може підтримувати актуальність інформації у своєму профілі, що є важливим для коректної роботи системи та комунікації з іншими користувачами або адміністраторами. Інтерфейс сторінки оформлений таким чином, щоб усі поля для редагування були легко доступними, а процес оновлення даних - максимально простим і зрозумілим.

Рис. 3.7. Модальне вікно для зміни паролю.

У вікні зміни паролю користувачу пропонується опція «Вийти на всіх інших пристроях». Ця функція виконує важливу роль у забезпеченні безпеки облікового запису. Якщо користувач активує цей пункт перед збереженням нового паролю, система автоматично завершить усі активні сесії на інших пристроях, з яких раніше було здійснено вхід до облікового запису. Винятком є лише поточний пристрій, з якого виконується зміна паролю.

|      |      |                   |        |      |                                         |      |
|------|------|-------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А. |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.    |        |      |                                         | 46   |
| Змн. | Арк. | № докум.          | Підпис | Дата |                                         |      |

Такий підхід дозволяє користувачу бути впевненим, що ніхто зі сторонніх осіб, навіть якщо раніше мав доступ до акаунту, більше не зможе ним скористатись без повторної авторизації. Ця функція особливо корисна в ситуаціях, коли є підозра, що доступ до акаунту могли отримати треті особи або коли користувач змінює пароль з міркувань безпеки.

Перейдемо на сторінку «Учні», щоб перейти на конкретного учня (рис. 3.8). Для цього натискаємо на відповідну кнопку в боковому меню.

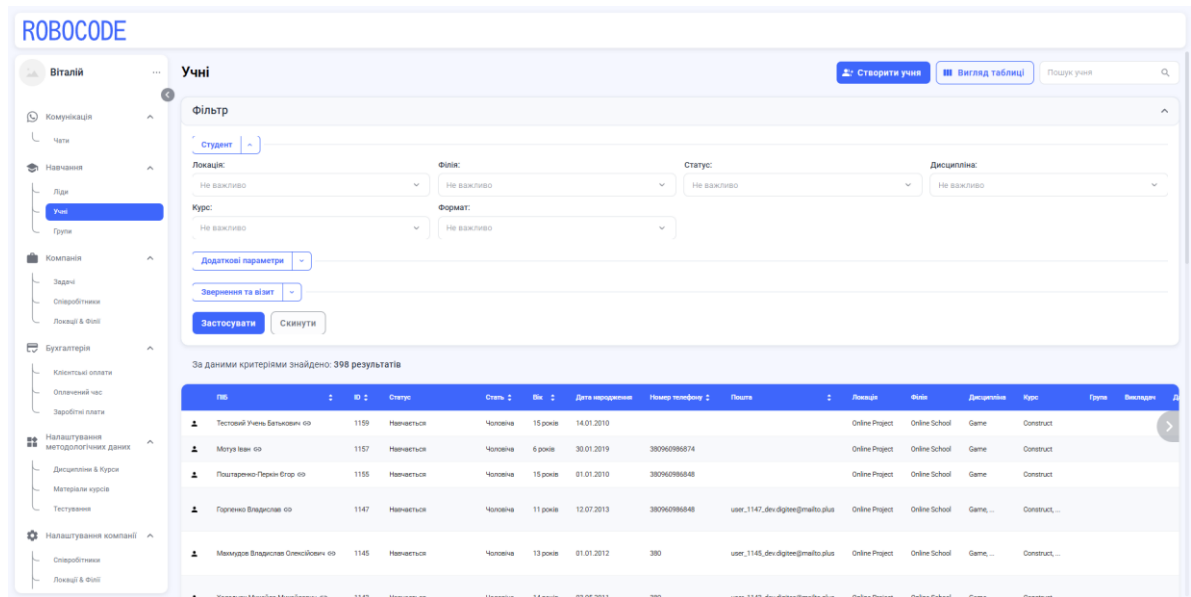


Рис.3.8. Сторінка учнів.

Починаючи з цієї сторінки і на всіх наступних, огляд зосереджено виключно на робочій області інтерфейсу, оскільки навігаційна панель та бокове меню залишаються статичними і не зазнають змін у візуальному чи функціональному плані. Це дозволяє сконцентрувати увагу на динамічному вмісті, який змінюється в залежності від розділу або дій користувача. На поточному екрані представлено таблицю з переліком учнів, яка має зручну пагінацію - це дає змогу розбити великий обсяг даних на окремі сторінки та не перевантажувати інтерфейс. Для зручності користувача реалізовано функцію фільтрації, яка дозволяє швидко знаходити потрібного учня за заданими критеріями. Варто зазначити, що сам фільтр є доволі гнучким і функціональним - він містить орієнтовно п'ятнадцять параметрів, за якими можна здійснювати пошук, що значно спрощує роботу зі списком у великих навчальних закладах. Крім того, у верхній частині таблиці

розміщені дві важливі кнопки: одна відповідає за створення нового учня, натискання на неї відкриває відповідну форму для введення персональних даних, інша дозволяє змінити вигляд самої таблиці, зокрема налаштувати, які саме поля повинні бути відображені. Це дає змогу адаптувати інтерфейс під конкретні потреби користувача, роблячи взаємодію із системою більш зручною та ефективною.

Перейдемо до конкретного учня, натиснувши на його ім'я в таблиці.

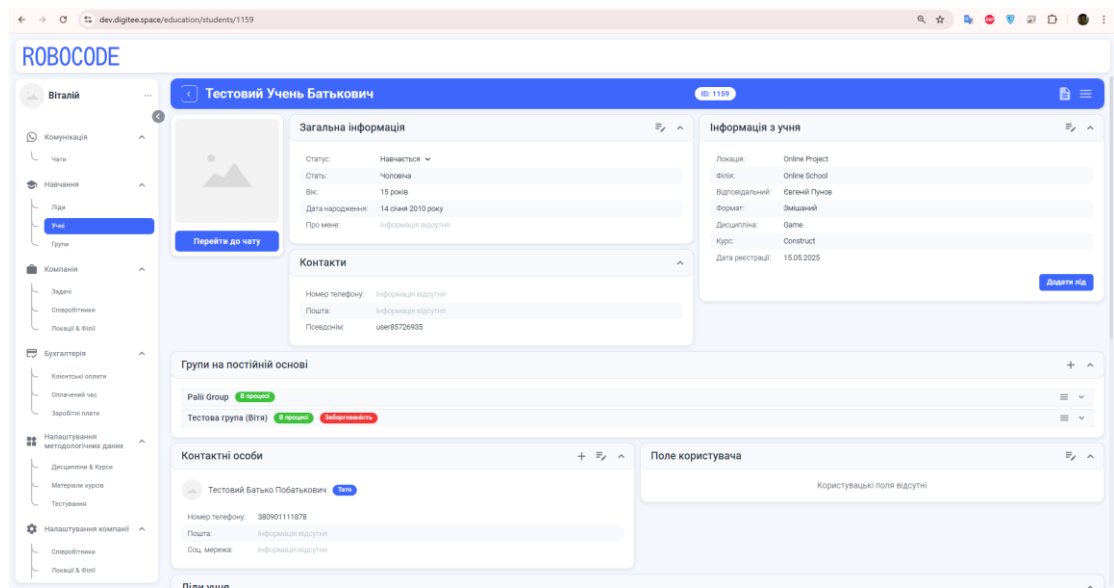


Рис.3.9. Сторінка учня.

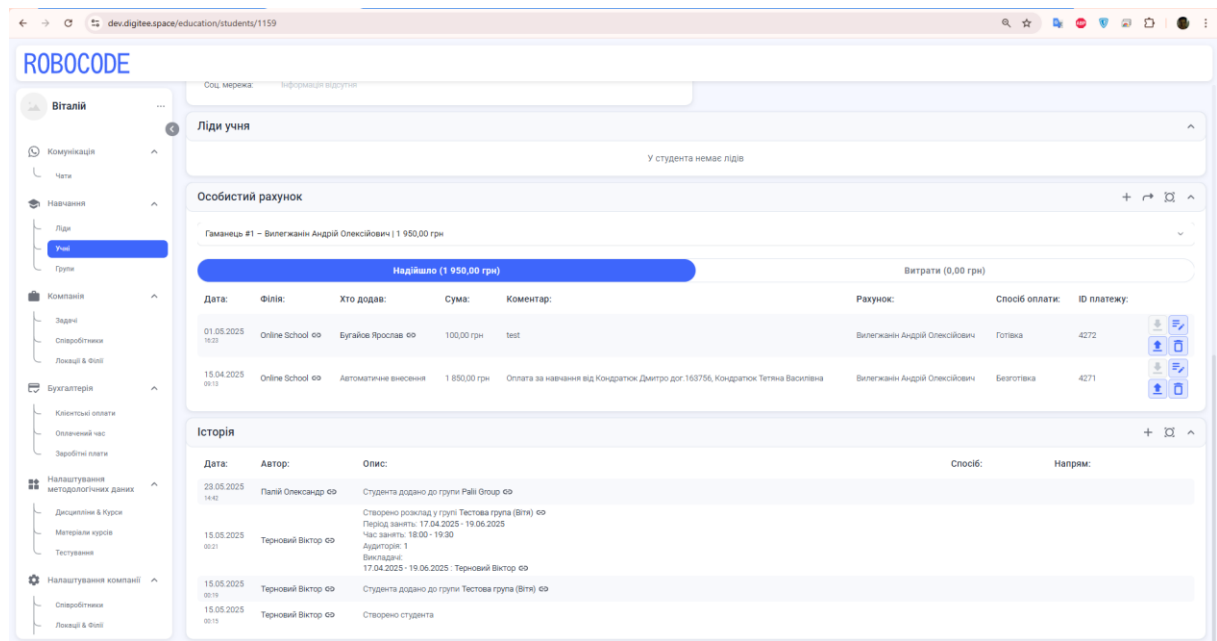


Рис.3.10. Сторінка учня.

На сторінці окремого учня представлено широкий набір інформаційних блоків, які дозволяють отримати повну картину щодо поточного стану учня у системі. У верхній частині сторінки знаходиться розділ із загальними даними, де відображено основну персональну інформацію - ПІБ, дата народження, контактні дані. Нижче розміщено блок, присвячений навчанню в групах. У ньому перелічено всі навчальні групи, в яких бере участь учень, із зазначенням дати зарахування, поточного статусу, викладача та інших деталей. Цей блок буде розглянуто нижче.

Також на сторінці присутній блок з фінансовою інформацією, зокрема з даними про особистий рахунок учня. Тут можна побачити актуальний баланс, зафіксовані платежі, заборгованість (якщо є), а також будь-які попередні транзакції. Поруч знаходиться хронологічний список дій, пов'язаних із цим учнем, а саме історія дій. Це дозволяє простежити зміну статусів, внесення змін до профілю, реєстрацію платежів та інші дії з боку адміністратора, викладачів або менеджерів.

У правому верхньому кутку сторінки розміщена кнопка з іконкою налаштувань. При натисканні на неї відкривається меню, в якому можна надати учневі персональний доступ до його облікового запису в системі. Це надзвичайно корисна функція, адже дозволяє учню самостійно входити до платформи для виконання тестових завдань, ознайомлення з навчальними матеріалами, перегляду розкладу тощо.

Крім цього, присутні функції редагування користувач із відповідними правами може оновити загальну інформацію про учня, а також додавати або видаляти контактних осіб. Усі ці можливості роблять сторінку учня багатофункціональним інструментом для управління навчальним процесом та супровідною інформацією.

Натиснувши на одну з груп, можна побачити детальну інформацію про навчання учня в цій групі (рис.3.11)

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         | 49   |

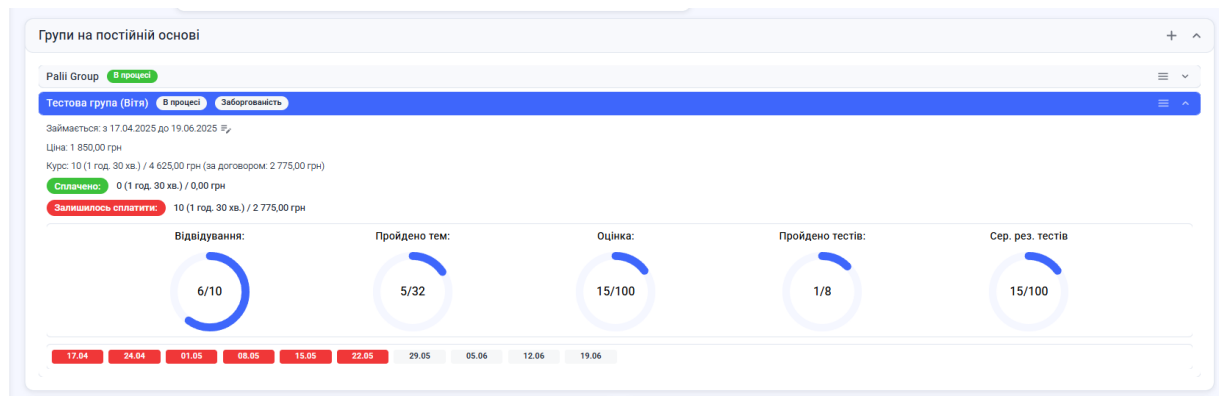


Рис.3.11. Детальна інформація про навчання учня в групі.

На сторінці з детальною інформацією про навчання учня в групі зосереджено всі ключові дані, необхідні для аналізу успішності, фінансових взаєморозрахунків та відвідуваності. У верхній частині представлено назву навчальної групи, після чого відображається основна інформація: вартість навчання в цій групі, кількість занять у повному курсі, кількість занять, які вже були оплачені, загальна сума коштів, що вже надійшли, та сума, яка ще підлягає оплаті. Це дозволяє оперативно контролювати фінансову ситуацію щодо конкретного курсу.

Окрім цього, присутній розділ зі стислою статистикою, що включає показники відвідуваності та загальної успішності учня. Вони базуються на внутрішній системі оцінювання та фіксації присутності, що дає змогу зручно аналізувати динаміку навчального процесу. Трохи нижче розміщений розклад занять, який дозволяє переглядати, коли саме відбуваються уроки, скільки вони тривають, хто з викладачів їх проводить.

У випадках, коли за учнем числиться заборгованість по оплаті навчання в групі, система автоматично повідомляє про це, розміщуючи попередження поруч із назвою групи. Крім того, такі заняття будуть позначені червоним кольором у розкладі, щоб мати змогу швидко звернути увагу на цю проблему. Завдяки цьому вдається уникнути непорозумінь і контролювати навчальний процес із урахуванням фінансових зобов'язань.

Також реалізовано можливість взаємодії з навчальними групами, для цього в правій частині блоку з групою є відповідні кнопки. Можна створити фінансову

|      |      |                   |        |      |                                         |      |
|------|------|-------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А. |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.    |        |      |                                         | 50   |
| Змн. | Арк. | № докум.          | Підпис | Дата |                                         |      |

операцію - зарахувати оплату за певну кількість занять. Можливо повернути залишок коштів на особовий рахунок учня, якщо така потреба виникла.

Зарахування коштів на особовий рахунок учня може здійснюватися двома способами. Перший - ручний, коли менеджер самостійно фіксує надходження коштів після того, як батьки учня оплатили. У такому разі менеджер відкриває профіль учня, вводить дані про платіж і додає відповідну суму на рахунок (рис. 3.12). Другий спосіб - автоматизований, при якому система самостійно обробляє платежі, здійснені через платіжні сервіси. Автоматизований процес обробки платежів буде розглянуто детальніше нижче.

Завдяки цій структурі сторінка надає зручний та функціонально насичений інтерфейс для управління не лише навчальним процесом, але й фінансовими аспектами, що значно полегшує щоденну роботу адміністративного персоналу та забезпечує прозорість для батьків і учнів.

Рис. 3.12. Модальне вікно для створення операції зарахування коштів.

Під час виконання операції зарахування коштів на особовий рахунок студента необхідно коректно заповнити форму, що з'являється у вигляді модального вікна. У цій формі передбачено кілька полів для заповнення, і деякі з них є обов'язковими для успішного завершення операції. Зокрема, обов'язковими є поля: тип оплати, спосіб оплати, сума, а також рахунок, на який мають бути

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         | 51   |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         |      |

зараховані кошти. Ці дані необхідно вказати точно, оскільки від цього залежить правильність відображення інформації в системі.

Після введення всієї необхідної інформації, наприклад, якщо вказати суму в розмірі 2000 гривень, натискання кнопки підтвердження призведе до автоматичного зарахування цієї суми на особовий рахунок обраного студента. Одразу після цього система виконує низку дій: оновлює баланс рахунку студента, додаючи до нього зазначену суму; у журналі фінансових операцій формується новий запис, який містить деталі операції — дату, час, спосіб оплати, суму та користувача, що здійснив зарахування.

Також у відповідному розділі особового рахунку з'являється цей запис, що дозволяє у будь-який момент перевірити історію змін і впевнитися в тому, що операція дійсно була виконана. Крім того, для зручності користувача система автоматично показує повідомлення про успішне завершення операції. Це повідомлення з'являється у правому верхньому куті екрану і містить коротку інформацію про зараховану суму та статус дії. Таким чином, користувач відразу отримує підтвердження, що всі необхідні дії були виконані вірно, і система зберегла інформацію про операцію (рис. 3.13).

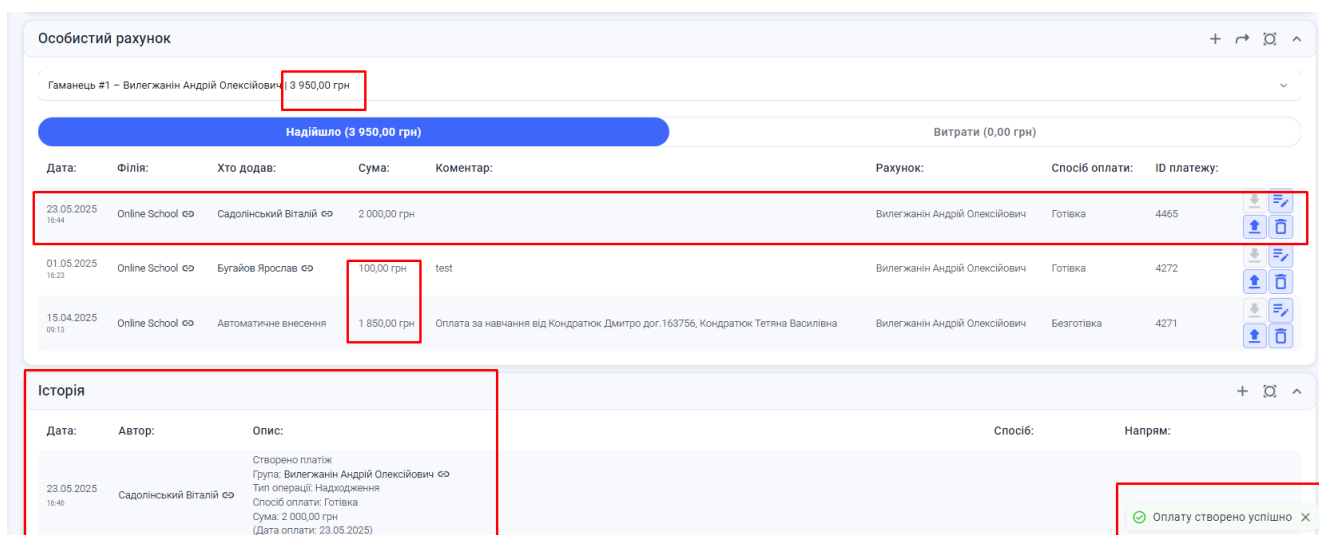


Рис.3.13. Успішне внесення оплати.

Система також надає зручний функціонал для перегляду всієї історії фінансових операцій, пов'язаних з особовим рахунком конкретного студента. Для

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         | 52   |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         |      |

того щоб скористатись цією функцією, необхідно натиснути на спеціальну іконку, яка розташована у правому верхньому куті інформаційного блоку з назвою «Особистий рахунок». Після натискання відкривається окреме вікно або розділ, в якому представлена таблиця з переліком усіх транзакцій, що коли-небудь виконувалися з цим рахунком (рис. 3.14).

| Дата              | Опис                               | Рахунок                        | Витрати      | Надходження  |
|-------------------|------------------------------------|--------------------------------|--------------|--------------|
| 27.05.2025        | Списання у Pail Group co           | Вилежанин Андрій Олександрович | 462.50 грн   |              |
| 27.05.2025        | Списання                           | Вилежанин Андрій Олександрович | 1 850.00 грн |              |
| 27.05.2025        | Списання у Pail Group co           | Вилежанин Андрій Олександрович | 1 850.00 грн |              |
| 27.05.2025        | Списання у Pail Group co           | Вилежанин Андрій Олександрович | 1 850.00 грн |              |
| 27.05.2025        | Надходження                        | Вилежанин Андрій Олександрович |              | 1 850.00 грн |
| 23.05.2025        | Списання у Тестова група (Вітк) co | Вилежанин Андрій Олександрович | 1 850.00 грн |              |
| 23.05.2025        | Надходження                        | Вилежанин Андрій Олександрович |              | 2 000.00 грн |
| 23.05.2025        | Надходження                        | Вилежанин Андрій Олександрович |              | 4 070.00 грн |
| 01.05.2025        | Надходження                        | Вилежанин Андрій Олександрович |              | 100.00 грн   |
| 15.04.2025        | Надходження                        | Вилежанин Андрій Олександрович |              | 1 850.00 грн |
| Загалом           |                                    |                                | 7 862.50 грн | 9 870.00 грн |
| Коштів на рахунок |                                    |                                |              | 2 007.50 грн |

Рис. 3.14. Операції з особовим рахунком.

У таблиці кожен запис містить інформацію про відповідну операцію: дату її проведення, тип операції (наприклад, зарахування, повернення, списання), суму а також рахунок. Це дозволяє повністю контролювати фінансову історію студента.

Додатковою перевагою цього розділу є автоматичне підбиття підсумків. Наприкінці таблиці з усіма операціями відображається загальний залишок коштів на рахунку. Це дозволяє з першого погляду дізнатись, скільки грошей наразі доступно без потреби вручну підраховувати суми окремих транзакцій. Завдяки цьому користувач отримує зручний та ефективний інструмент для моніторингу фінансів студента в рамках системи.

Тепер з достатньою кількістю коштів на рахунку перейдемо до оплати занять (рис.3.15).

Рис.3.15. Модальне вікно оплати занять.

Після заповнення форми оплати у модальному вікні користувач має змогу одразу побачити, скільки занять буде оплачено на підставі введеної суми або ж кількості абонементів. Це відображається автоматично в нижній частині форми і оновлюється динамічно при зміні даних. Якщо для обраної групи встановлено можливість часткової оплати, система дозволяє вручну вказати або конкретну суму, яку користувач бажає оплатити, або точну кількість абонементів, за які має бути здійснено платіж.

Окрім цього, форма надає додаткові можливості для врахування фінансових коригувань. Зокрема, з правого боку від поля, в якому вказується кінцева сума до сплати, розміщені кнопки для додавання знижок або доплат. При натисканні на ці кнопки відкривається можливість ввести відповідні значення, які будуть автоматично враховані при розрахунку остаточної суми. Це дозволяє адаптувати платіж до конкретної ситуації, наприклад, застосувати знижку для учня, який має пільги, або додати доплату.

У випадку внесення змін до полів, що стосуються суми або кількості абонементів, система миттєво оновлює інформацію про те, скільки занять буде фактично оплачено. Це забезпечує користувача максимально точними даними до моменту завершення транзакції, дозволяючи уникнути непорозумінь і помилок.

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолинський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         | 54   |

Після підтвердження оплати, якщо раніше існувала заборгованість за відвідування занять, її розмір автоматично зменшується відповідно до сплаченої суми. Водночас, у загальному підсумку зростає обсяг коштів, що були внесені за весь період навчання у цій групі. Всі ці зміни зберігаються у системі, та можуть бути використані для подальшого фінансового аналізу або формування звітності (рис. 3.16). Таким чином, процес оплати є максимально прозорим, контрольованим і зручним для користувача.

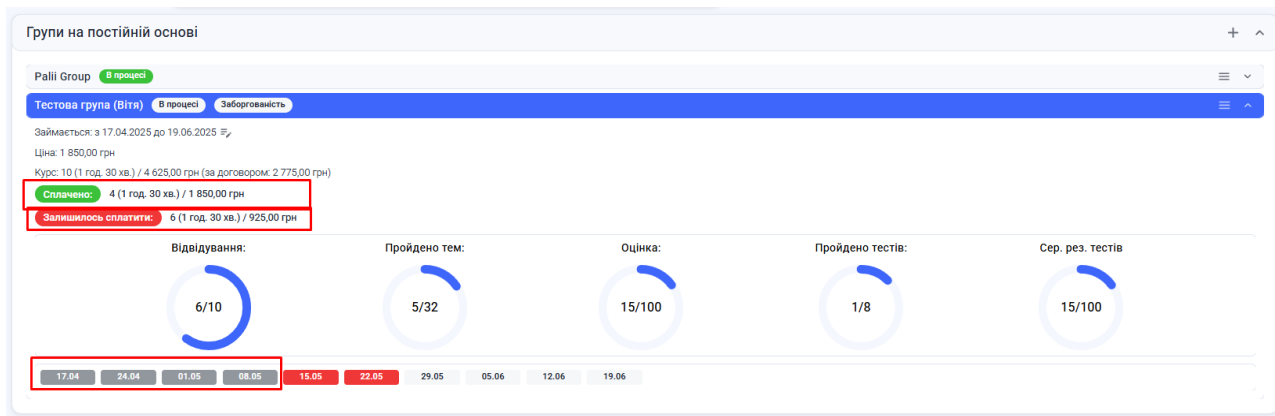


Рис.3.16. Успішно проведена оплата занять в групі.

Відповідні записи про оплату, з’являться в історії та в списку операцій (рис. 3.17).

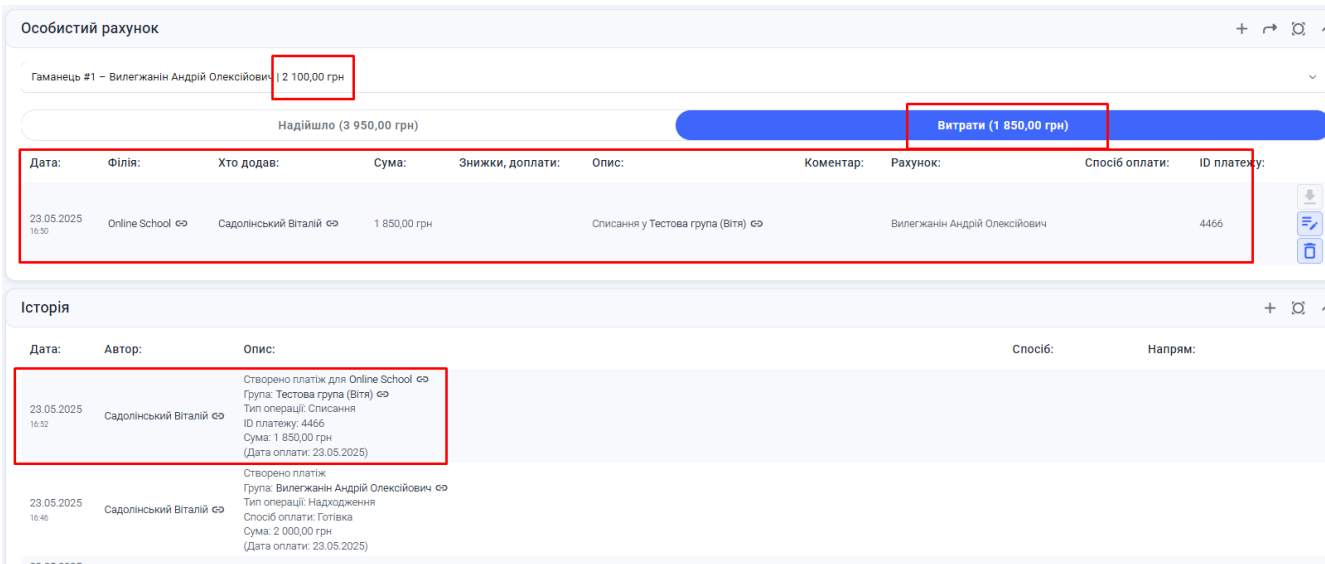


Рис. 3.17. Записи про оплату занять.

Для більш детального перегляду історії можна перейти на сторінку повної історії дій, які відбувались з учнем (рис. 3.18).

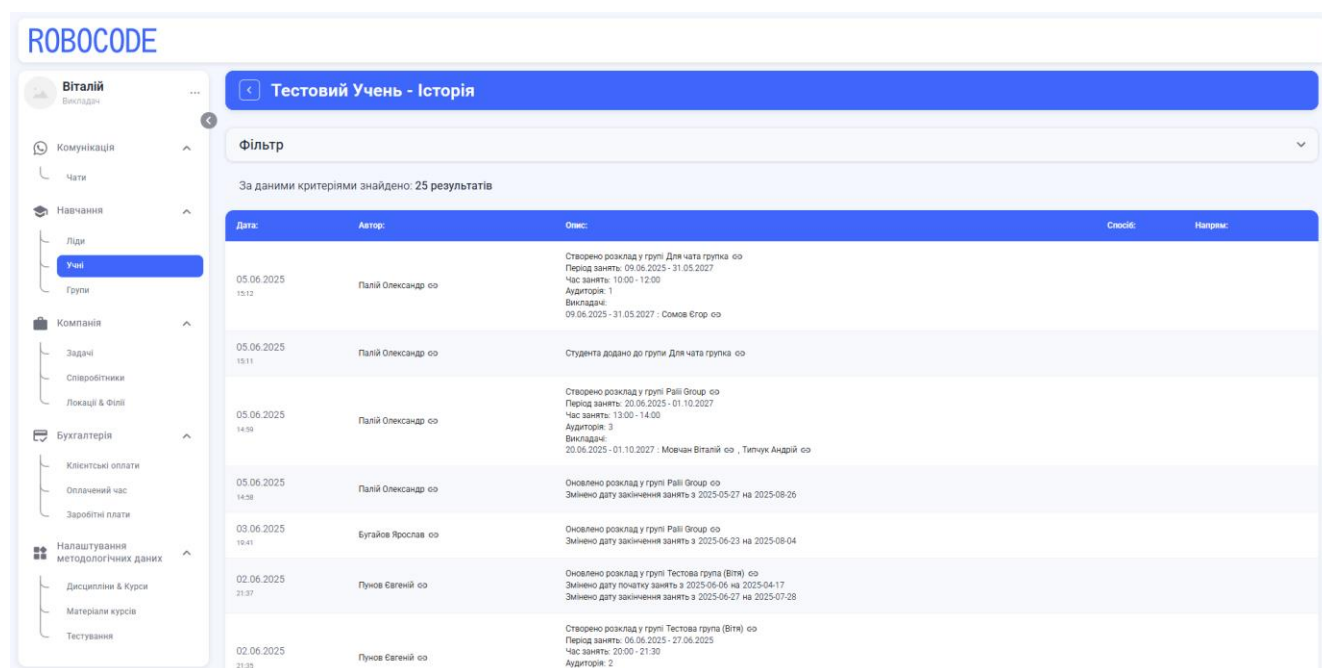


Рис. 3.18. Сторінка повної історії дій з користувачем.

На представленій сторінці користувач має можливість ознайомитися з повною історією дій, що були здійснені з поточним студентом у межах системи. Уся інформація подається у вигляді впорядкованого списку, який містить детальний перелік усіх подій, пов'язаних з учнем. Це можуть бути як адміністративні зміни (наприклад, оновлення особистих даних, додавання або видалення контактних осіб), так і дії, пов'язані з навчальним процесом (зарахування до групи, зміна статусу участі, перегляд матеріалів тощо).

Важливою функціональною можливістю цієї сторінки є наявність зручного фільтра, розташованого у верхній частині робочої області. Завдяки цьому фільтру можна швидко знайти потрібну інформацію, відсортувавши записи за обраними критеріями. Наприклад, можна переглядати лише певний тип подій або ж події за вказаний період часу. Це значно полегшує роботу адміністрації, дозволяючи оперативно аналізувати зміни, що відбувалися з обліковим записом конкретного користувача.

Також кожен запис в історії містить точну дату та час, а також зазначення дії, яка була виконана, і суб'єкта, що її ініціював. Це дозволяє забезпечити високу

ступінь прозорості та сприяє підвищенню відповідальності з боку користувачів, які мають доступ до редагування даних у системі.

Перейдемо до блоку сторінок з налаштуваннями параметрів та сутностей потрібних у роботі CRM-системи (рис.3.19)

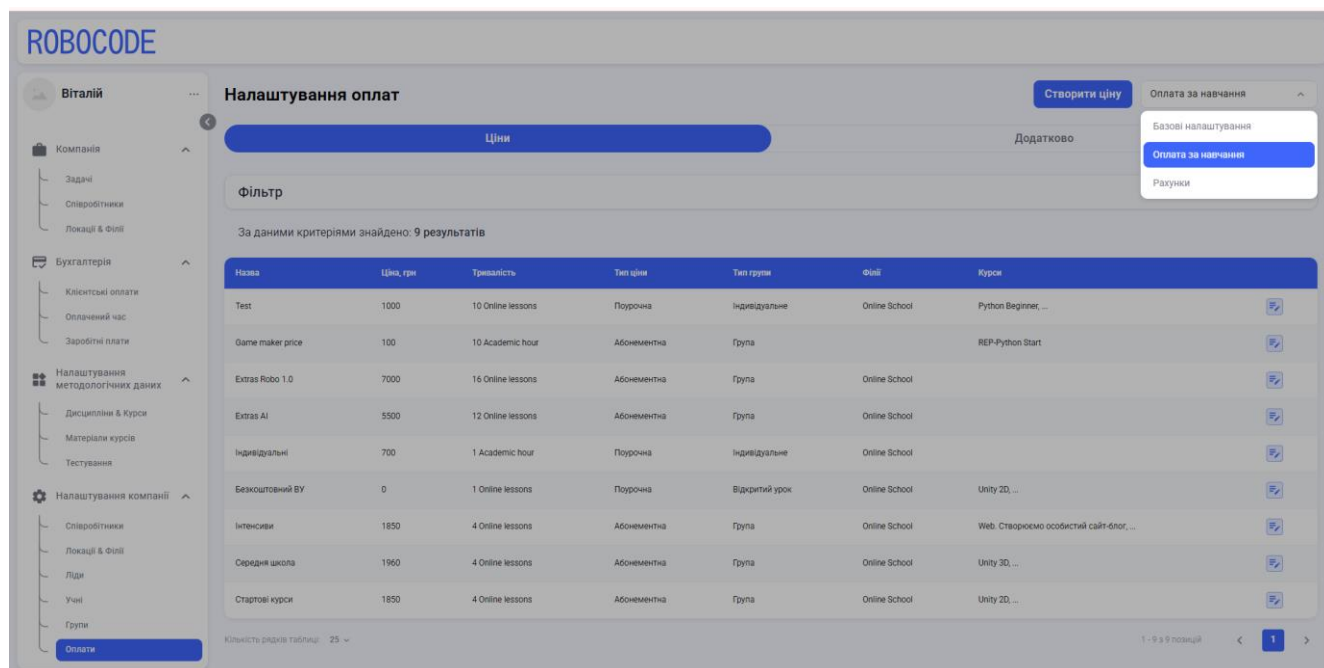


Рис. 3.19. Сторінка налаштувань.

На цій сторінці реалізовано функціонал, який дозволяє виконувати налаштування ключових параметрів, що впливають на роботу системи. Для зручної навігації по розділу налаштувань передбачено контекстне меню. Це меню розташовано у доступному місці інтерфейсу та дозволяє легко перемикатись між різними підрозділами налаштувань. Таким чином, користувач без додаткових зусиль може швидко знайти потрібну категорію та внести необхідні зміни.

Перейдемо до сторінки з налаштування рахунків (рис. 3.20), яка відіграє важливу роль у загальній конфігурації фінансового модуля системи. На цій сторінці здійснюється налаштування рахунків, налаштування полів для синхронізації з банківськими системами, перевірка статусу інтеграції, також є можливість зробити тестовий запит до API банківської системи.

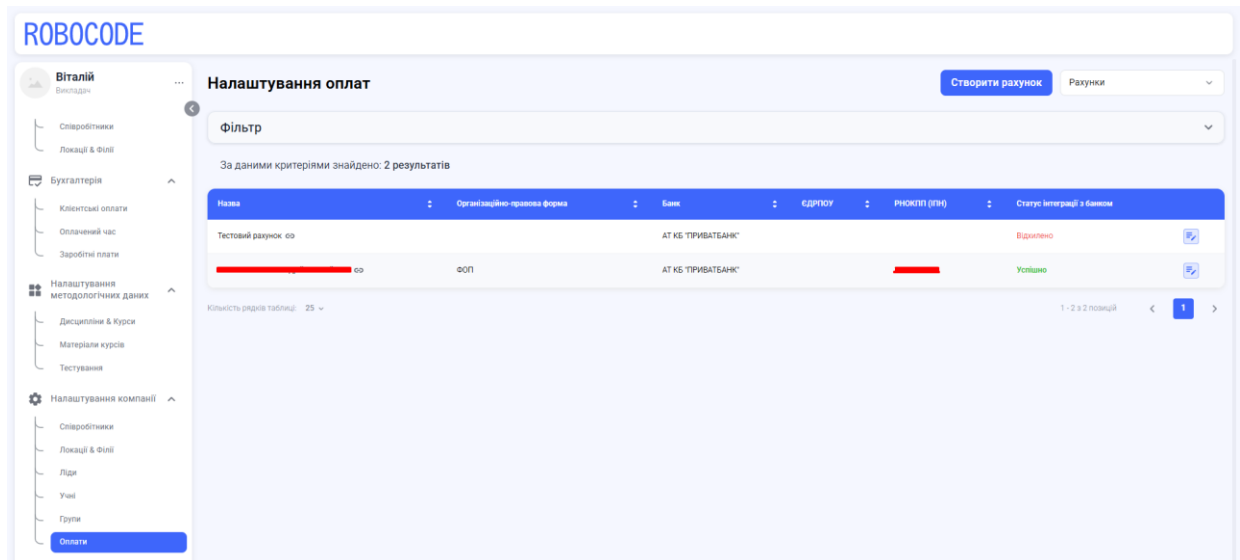


Рис. 3.20. Сторінка налаштування рахунків.

На даній сторінці представлений повний перелік банківських рахунків, які були попередньо додані до системи та використовуються для синхронізації фінансових операцій. Кожен рахунок відображається у вигляді окремого рядка з усією необхідною інформацією. Це дозволяє адміністраторам швидко переглядати стан рахунків, перевіряти їх актуальність та виявляти можливі проблеми із синхронізацією.

Поруч із кожним рахунком розміщена кнопка, натиснувши на яку відкривається форма редагування даних поточного банківського рахунку. Це особливо корисно у випадках, коли виникає потреба оновити інформацію після зміни банківських реквізитів.

Особливу увагу варто звернути на колонку, яка має назву «Статус інтеграції з банком». Цей статус показує, чи вдалося системі встановити зв'язок із банківським API та отримати актуальні дані про платежі. Після кожної синхронізації з банком це поле оновлюється, і користувач бачить, чи була передача даних успішною. Таким чином, можна оперативно виявити та усунути проблеми з інтеграцією.

Крім того, доступна можливість виконання тестового запиту до банку без запуску повної синхронізації. Такий запит дозволяє перевірити працездатність інтеграції з конкретним рахунком. Для його виконання потрібно відкрити модальне вікно редагування рахунку, де розміщено кнопку для ініціації тестового

|      |      |                   |        |      |                                         |      |
|------|------|-------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А. |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.    |        |      |                                         |      |
| Змн. | Арк. | № докум.          | Підпис | Дата |                                         | 58   |

звернення до банківського сервісу (рис. 3.21). У результаті користувач отримує оперативний зворотний зв'язок щодо стану підключення, що дозволяє контролювати технічну справність налаштувань ще до моменту обробки реальних транзакцій.

Рис. 3.21. Модальне вікно редагування банківського рахунку.

Щоб отримати можливість перевірити підключення до банківської системи за допомогою тестового запиту, користувач повинен попередньо заповнити обов'язкові поля, що стосуються автентифікаційних даних. До таких полів належать «token» та «clientId». Ці дані надаються банком при налаштуванні інтеграції та є необхідними для встановлення захищеного з'єднання між системою та банківським API. Без заповнення цих полів система не зможе створити авторизований запит, а відповідна функція буде недоступною.

Після внесення правильних значень у поля «token» та «clientId» кнопка «Тестовий запит до банку», стає активною, тобто її можна натискати. Це означає, що система підготувалась до надсилання запиту та має достатньо інформації для його виконання.

Після натискання на зазначену кнопку відбувається надсилання запиту до банківського сервера. У відповідь користувач отримає результат тестування у

|      |      |                   |        |      |                                         |      |
|------|------|-------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А. |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.    |        |      |                                         | 59   |
| Змн. | Арк. | № докум.          | Підпис | Дата |                                         |      |

вигляді повідомлення, яке відображатиме поточний статус підключення. Це може бути як успішна відповідь, яка підтверджує коректність з'єднання та даних, так і повідомлення про помилку у разі невірних параметрів або технічних збоїв. Таким чином, дана функція дозволяє оперативно перевірити працездатність інтеграції ще до повноцінного використання банківського рахунку у реальних операціях. Це суттєво знижує ризик виникнення критичних помилок у роботі з фінансами.

Перейдемо до сторінки синхронізації оплат з банківською системою (рис.3.22).

ROBOCODE

Віталій

...

Ліди

Учні

Групи

Компанія

Задані

Співробітники

Локації & Файли

Бухгалтерія

Календар оплат

Опалений час

Зарплатні платіж

Налаштування методологічних даних

Дисципліни & Курси

Матеріали курсів

Тестування

Налаштування компанії

Співробітники

Локації & Файли

Ліди

Учні

Групи

Оплати

Клієнтські оплати

Синхронізувати оплати

Вигляд таблиці

Фільтр

За даними критеріями знайдено: 2364 результатів

| Дата платежу     | Сума оплати, грн | Платіж            | Розрах. | Об'єкт оплати                | Фінанс. | Локація       | Контрагент                                                                                                                                                              | Хто здійснив платіж  | Статус   |
|------------------|------------------|-------------------|---------|------------------------------|---------|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------|----------|
| 23.05.2025 16:44 | 2 000,00         | Тестовий Учень    | со      | Виплати Андрій Олександрович | со      | Online School | Online Project                                                                                                                                                          | Садолінський Віталій | Готова   |
| 23.05.2025 13:00 | 4 070,00         | -                 | со      | Виплати Андрій Олександрович | со      | Online School | Online Project<br>Оплата за навчання, ІД дитини 184955, Трудовикова Поліна, Трудовикова Наталя Олександрівна                                                            | Автоматичне внесення | Безготів |
| 23.05.2025 12:28 | 1 214,00         | -                 | со      | Виплати Андрій Олександрович | со      | Online School | Online Project<br>Оплата за навчання Севаст'янова Ірини, Севаст'янова Лілія Григорівна                                                                                  | Автоматичне внесення | Безготів |
| 23.05.2025 12:06 | 600,00           | -                 | со      | Виплати Андрій Олександрович | со      | Online School | Online Project<br>Оплата за навчання під Прикладно Вікторія Віталівна, Прикладно Вікторія Віталівна                                                                     | Автоматичне внесення | Безготів |
| 23.05.2025 11:32 | 2 290,00         | -                 | со      | Виплати Андрій Олександрович | со      | Online School | Online Project<br>Оплата за навчання дитини Соколовського Андрія Олександровича, ІД код 182122 від Соколовський Олександр Андрійович, Соколовський Олександр Андрійович | Автоматичне внесення | Безготів |
| 23.05.2025 09:40 | 1 862,00         | Данюк Олег        | со      | Виплати Андрій Олександрович | со      | Online School | Online Project<br>ІД 456 Данюк Олег Сергійович                                                                                                                          | Автоматичне внесення | Безготів |
| 20.05.2025 13:54 | 1,00             | Мамудов Владислав | со      | Виплати Андрій Олександрович | со      | Online School | Online Project                                                                                                                                                          | Садолінський Віталій | Готова   |
| 15.05.2025 14:49 | 1 758,00         | -                 | со      | Виплати Андрій Олександрович | со      | Online School | Online Project<br>Оплата за навчання під Шевченко Альона Крістіна, Шевченко Альона Крістіна                                                                             | Автоматичне внесення | Безготів |
| 14.05.2025 21:16 | 1 850,00         | Холдун Михайло    | со      | Виплати Андрій Олександрович | со      | Online School | Online Project<br>Оплата за навчання Рудковського Олександра Руслановича, ІД 236885, Максименко Ольга Василівна                                                         | Автоматичне внесення | Безготів |

Рис. 3.22. Сторінка синхронізації оплат.

На даній сторінці відображено всі платежі, які були автоматично додані до системи в результаті синхронізації з банком. У списку для кожної оплати виводиться основна інформація: дата платежу, сума, а також додаткові параметри. Праворуч біля кожного запису розміщені кнопки, які дозволяють відредагувати або повністю видалити відповідну операцію, якщо, наприклад, вона була внесена помилково або потребує уточнення.

Іноді система не може автоматично прив'язати платіж до конкретного студента. Найчастіше це трапляється у випадках, коли під час заповнення реквізитів до оплати особа, яка здійснює платіж, вказує некоректний або взагалі відсутній ідентифікатор студента. Наприклад, у полі «Призначення платежу»

|      |      |                   |        |      |                                         |      |
|------|------|-------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А. |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.    |        |      |                                         | 60   |
| Змн. | Арк. | № докум.          | Підпис | Дата |                                         |      |

може бути написано неправильний ID-студента, якого мав би стосуватись платіж. У такому випадку система не знає, на чий рахунок зараховувати кошти, і залишає запис без призначеного платника.

Щоб виправити цю ситуацію, адміністратор або відповідальний менеджер повинен вручну відредагувати запис про платіж і встановити правильного платника. Після того, як система отримає цю інформацію, кошти автоматично потрапляють на особовий рахунок відповідного студента. Якщо платника змінити повторно, то сума платежу спершу буде знята з рахунку попереднього користувача, а потім автоматично зарахована на рахунок нового. Це забезпечує правильний облік фінансів у системі.

Також на цій сторінці присутня кнопка, яка дозволяє запустити синхронізацію оплат з банківськими рахунками. При натисканні на цю кнопку з'являється спеціальне модальне вікно, в якому користувачеві необхідно вказати початкову і кінцеву дату періоду, за який потрібно оновити інформацію (рис. 3.23). Після підтвердження, система надсилає відповідні запити до банків і отримує всі дані про нові надходження. Таким чином, забезпечується актуальність інформації про платежі без необхідності ручного введення.

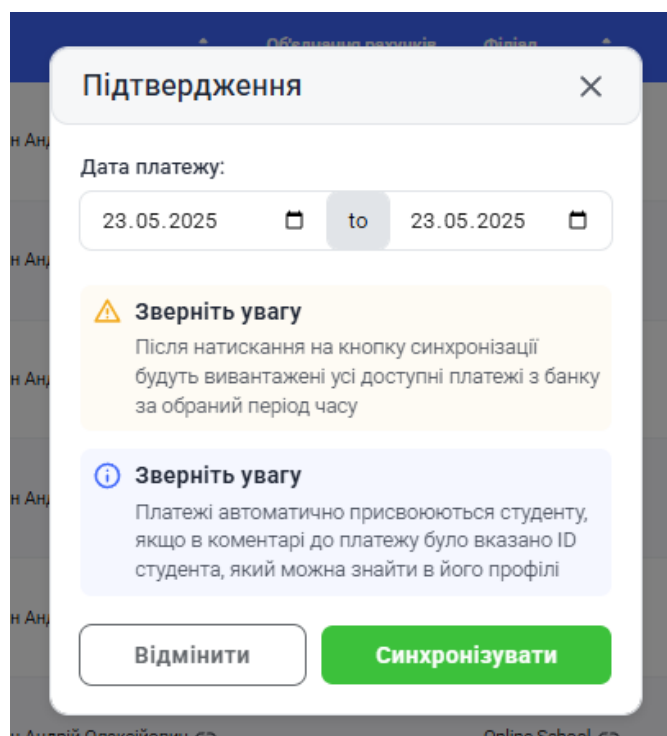


Рис. 3.23. Модальне вікно синхронізації оплат.

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         | 61   |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         |      |

Далі розглянемо один із важливих розділів системи, який стосується автоматизованого розрахунку заробітної плати викладачам. Щоб система могла коректно здійснювати нарахування, необхідно виконати певну послідовність дій.

Насамперед потрібно забезпечити, щоб викладач був прив'язаний до конкретної навчальної групи.

Однак, перед тим як додати викладача до групи, потрібно виконати ще один важливий підготовчий етап. Система потребує попереднього налаштування структури заробітної плати. Це включає створення категорій, підкатегорій, а також встановлення відповідних ставок оплати для кожної з них. Для цього в системі передбачена окрема сторінка налаштувань, де адміністратор може задати всі необхідні параметри. Саме на цій сторінці вводяться назви категорій, вказується, які саме підкатегорії до них відносяться, а також задаються числові значення ставок, що будуть використовуватись під час розрахунку зарплати.

Після цього, коли викладач буде доданий до групи, система автоматично врахує всі попередні налаштування, що значно спрощує подальшу обробку інформації.

Отже, для запуску процесу нарахування заробітної плати необхідно не просто додати викладача до навчального процесу, але й попередньо підготувати відповідну конфігурацію, яка забезпечить точність і прозорість у фінансових розрахунках. Тому зараз ми перейдемо до розгляду сторінки з налаштуваннями категорій, де безпосередньо здійснюється ця підготовка (рис. 3.24).

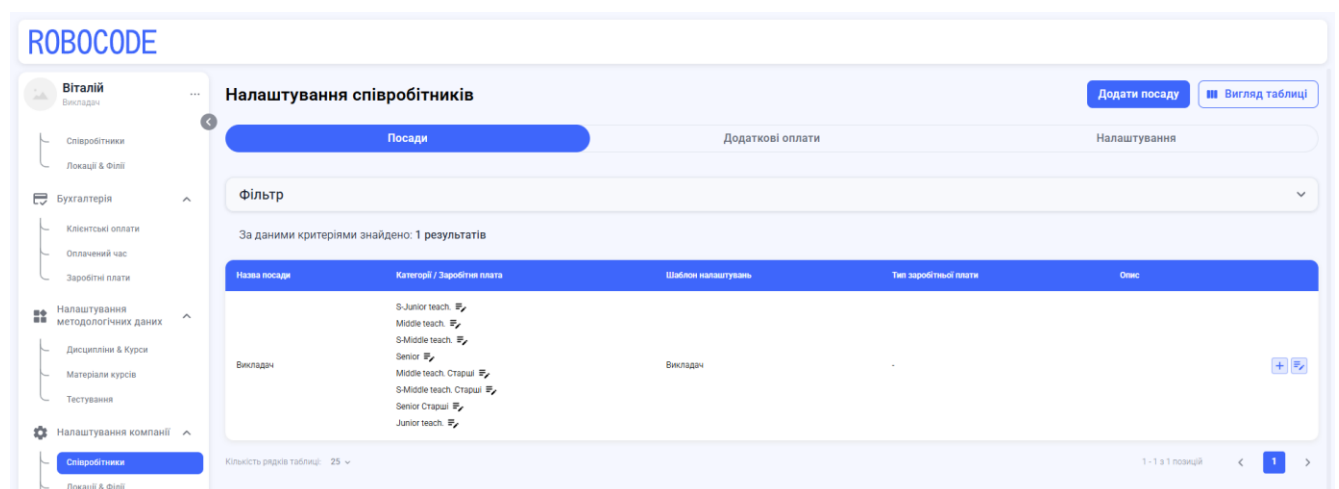


Рис. 3.24. Сторінка налаштування посад та категорій.

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         | 62   |

На цій сторінці користувач має можливість переглянути повний перелік усіх посад, які використовуються в системі. Для кожної з посад, у зручному вигляді відображається перелік категорій, які безпосередньо пов'язані з цією посадою. Це можуть бути різні напрямки, за якими здійснюється подальше нарахування заробітної плати. Наприклад, для викладача можуть існувати окремі категорії для групових занять, індивідуальних, підготовки навчальних матеріалів, кураторської діяльності тощо.

Інтерфейс побудований таким чином, щоб можна було швидко зорієнтуватись у структурі прив'язаних категорій до кожної посади. Це дозволяє адміністратору системи легко керувати налаштуваннями, пов'язаними з оплатою праці. Якщо потрібно внести зміни до вже існуючої категорії або створити нову, достатньо натиснути на відповідну кнопку, яка знаходиться поруч з назвою категорії. Після цього відкриється модальне вікно, яке дає змогу редагувати параметри поточної категорії.

У модальному вікні можна вказати назву категорії, обрати пов'язану з нею посаду, додати опис або інші додаткові дані, що будуть використовуватись при подальших розрахунках заробітної плати. Такий підхід дозволяє гнучко налаштовувати структуру оплати в залежності від типу діяльності, складності роботи та інших факторів. Отже, дана сторінка відіграє ключову роль у конфігурації системи оплати праці та забезпечує її гнучкість і адаптивність під різні моделі роботи в освітньому закладі. Після внесення змін усі нові параметри автоматично враховуються системою при наступних розрахунках.

Рис. 3.25. Модальне вікно для створення та редагування категорій.

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         | 63   |

В поточному вікні можна побачити такі поля як: назва категорії, тип заробітної плати, залежність заробітної плати та список підкатегорій. Ці поля потрібні для налаштування фінансової логіки на рівні конкретних. Наприклад, тип заробітної плати визначає, чи буде оплата погодинною, фіксованою або залежатиме від кількості занять. Залежність заробітної плати дозволяє встановити правила обчислення на основі кількості учнів у групі або ж кількості учнів на занятті. Список підкатегорій забезпечує гнучкість у створенні складних фінансових структур - наприклад, окрема категорія для викладачів з надбавками.

Натиснувши на іконку редагування підкатегорії ми перейдемо до відповідної сторінки редагування (рис. 3.26).

Рис. 3.26. Сторінка налаштування підкатегорій.

На поточній сторінці користувач має можливість налаштувати підкатегорії оплати праці, які відповідають за точне розрахування заробітної плати в залежності від умов проведення занять. У верхній частині інтерфейсу представлений список усіх доступних підкатегорій, серед яких можна обрати ту, яку потрібно налаштувати. Після вибору підкатегорії відкривається можливість додавання одного або кількох блоків налаштувань, які описують конкретні умови оплати за заняття. У кожному такому блоці потрібно вказати тип групи -

наприклад, індивідуальна чи групова форма навчання, а також її формат - онлайн чи офлайн.

Крім цього, обов'язково зазначається кількість учнів, що є важливим фактором при розрахунку ставки, адже для більшої кількості учнів може передбачатися інша система нарахувань. Далі вноситься конкретна сума, яка буде автоматично нарахована викладачеві за проведене заняття в таких умовах. Таким чином, кожен блок дозволяє системі точно визначити суму, яку потрібно виплатити за певний тип навчального навантаження.

Система підтримує додавання великої кількості таких блоків для кожної підкатегорії, що дозволяє з високою точністю і гнучкістю враховувати найрізноманітніші сценарії, які можуть виникати в реальному освітньому процесі.

Це особливо корисно для шкіл, де працює багато викладачів і де заняття проходять у різних форматах і з різним навантаженням. Такий підхід забезпечує прозорість і справедливість при нарахуванні заробітної плати.

Окремо варто відзначити блок із налаштуванням значень за замовчуванням. Цей блок відіграє резервну роль - у випадку, коли система з якихось причин не зможе підібрати відповідні налаштування на основі параметрів групи, вона використовуватиме саме ці значення. Це дозволяє уникнути ситуацій, коли викладачу не буде нараховано оплату через відсутність відповідного блока. Завдяки цьому забезпечується стабільність роботи системи та мінімізуються людські помилки при призначенні викладачів.

Після завершення налаштування підкатегорій та блоків оплати, користувач може перейти до наступного етапу, а саме процесу виплати заробітної плати. Для цього потрібно перейти у відповідний розділ під назвою «Співробітники», де відбувається подальше управління фінансовими операціями, пов'язаними з персоналом навчального закладу (рис. 3.27).

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         | 65   |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         |      |

ROBOCODE

Віталій

Комунікація

Чат

Навчання

Ліди

Учні

Групи

Компанія

Задчі

Співробітники

Локації & Файл

Бухгалтерія

Клієнтські оплати

Оплатений час

Заробітні плати

Налаштування методологічних даних

Дисципліни & Курси

Матеріали курсів

Тестування

Налаштування компанії

Співробітники

Локації & Файл

Співробітники

Додати співробітника

Вигляд таблиці

Пошук співробітника

Фільтр

За даними критеріями знайдено: 35 результатів

| ІД | ПІБ                              | Робочий статус | Стать  | Вік     | Дата народження | Повний телефон | Пошта        | Локація                     | Освітня форма  | Дата старту роботи | Роль       | Посади                                                     | Категорія |
|----|----------------------------------|----------------|--------|---------|-----------------|----------------|--------------|-----------------------------|----------------|--------------------|------------|------------------------------------------------------------|-----------|
|    | Палій Олександр                  | 1161           | Працює | Чоловік | 28 років        | 09.06.1996     | 380935396158 | sau222222@gmail.com         | Online Project | Online School      | 01.01.2018 | Адміністратор<br>Менеджер<br>Викладач<br>Офісний працівник | Викладач  |
|    | Гайдучен Максим Андрійович       | 1154           | Працює | Чоловік | 20 років        | 28.05.2004     | 380954839389 | maxim.gayduchen@gmail.com   | Online Project | Online School      | 08.04.2025 | Адміністратор<br>Менеджер<br>Викладач<br>Офісний працівник | Викладач  |
|    | Тест дублювати                   | 1152           | Працює | Чоловік | 21 рік          | 18.01.2004     | 380960986848 | Gfadedolski@gmail.com       | Online Project | Online School      | 31.03.2025 | Менеджер                                                   | Викладач  |
|    | Тест Сторінки Дублювати          | 1151           | Працює | Чоловік | 21 рік          | 18.01.2004     | 380960986848 | Vladolski@gmail.com         | Online Project | Online School      | 31.03.2025 | Адміністратор                                              | Викладач  |
|    | Каленін Деніс                    | 1150           | Працює | Чоловік | 34 років        | 29.08.1990     | 380660776023 | den-denis.kalakin@gmail.com | Online Project | Online School      | 01.11.2024 | Адміністратор<br>Менеджер<br>Викладач<br>Офісний працівник | Викладач  |
|    | Ододітський Віталій Анатолійович | 161            | Працює | Чоловік | 21 рік          | 18.01.2004     | 380960986848 |                             | Online Project | Online School      | 20.01.2025 | Адміністратор<br>Офісний працівник<br>Викладач<br>Менеджер | Викладач  |
|    | Бугайов Ярослав                  | 1149           | Працює | Чоловік | 22 років        | 04.09.2002     | 380992590457 | bugayov576@gmail.com        | Online Project | Online School      | 01.09.2024 | Адміністратор<br>Менеджер<br>Викладач<br>Офісний працівник | Викладач  |
|    | Сербін Олександр Сергійович      | 1108           | Працює | Чоловік | 26 років        | 21.06.1998     | 380951741229 | serbin.sasha1@gmail.com     | Online Project | Online School      | 04.03.2025 | Адміністратор<br>Офісний працівник                         | Викладач  |

Рис. 3.27. Сторінка співробітники.

На поточній сторінці представлено список співробітників, який відображається з використанням пагінації для зручності навігації серед великої кількості записів. Для швидкого пошуку потрібного співробітника передбачено фільтр, який за функціоналом дуже схожий на той, що використовується для пошуку учнів. Цей фільтр дозволяє відбирати співробітників за різними критеріями, що значно спрощує процес пошуку.

Також на сторінці розташована кнопка для додавання нового співробітника до системи, що дозволяє розширювати базу персоналу без зайвих труднощів.

Крім того, присутня кнопка для зміни вигляду таблиці, що дає можливість користувачу обрати найбільш зручний спосіб відображення інформації. Важливо зазначити, що ця сторінка є проміжною та використовується переважно для того, щоб швидко знайти конкретного співробітника і перейти до його детальної інформації або до розділу, де здійснюється нарахування та виплата заробітної плати (рис. 3.28).

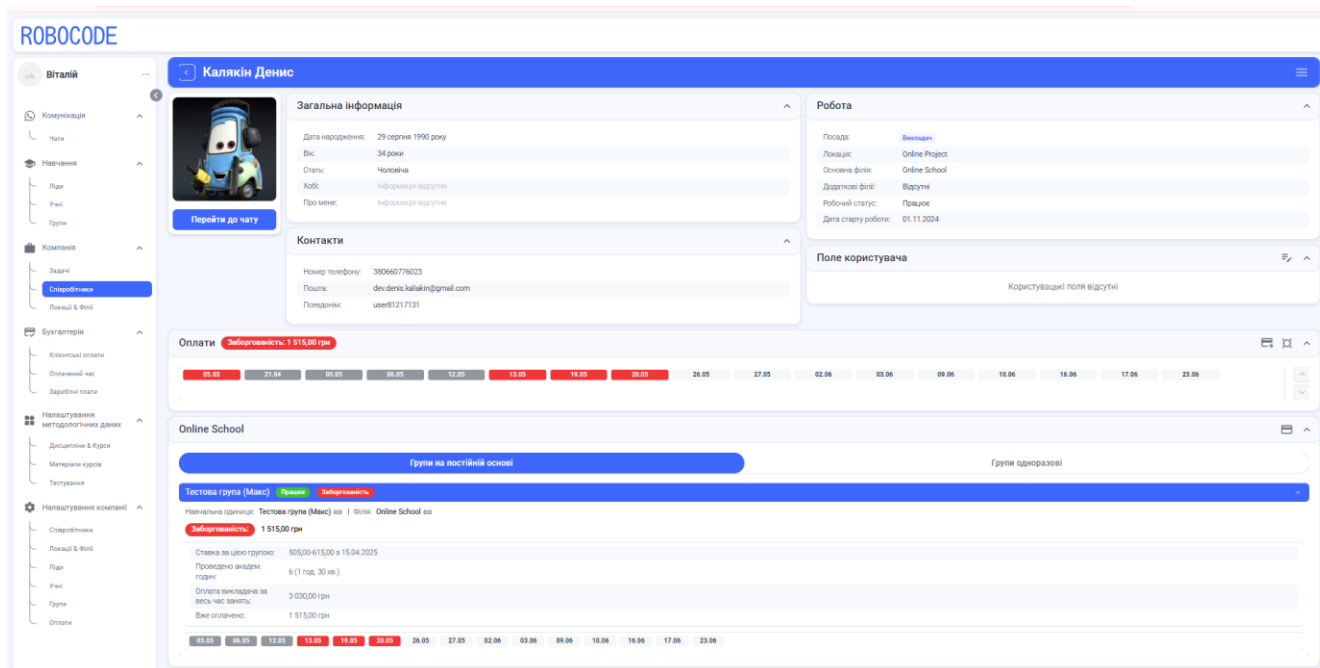


Рис.3.28. Сторінка викладача.

На цій сторінці користувач має змогу ознайомитися не лише із загальними відомостями про працівника, такими як його ім'я, посада, але й отримати інформацію про групи, до яких він залучений.

Для кожної групи, в якій працює працівник, відображається відповідний запис. Якщо натиснути на будь-яку з цих груп, відкривається детальніший огляд, який містить розширену інформацію про конкретну групу (рис. 3.29)..



Рис. 3.29. Детальна інформація про групу.

Серед представленої інформації можна детально побачити кількість коштів, яка призначена для виплати викладачу за роботу з конкретною групою. Окрім цього, відображається ставка, що була призначена для цієї групи, а також кількість занять, які вже проведені в межах навчального процесу. Уточнюється

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         | 67   |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         |      |

також загальна сума коштів, яка була оплачені викладачу, і скільки залишилося виплатити відповідно до нарахованої заробітної плати. Ця інформація не є унікальною для сторінки працівника, вона також дублюється і відображається на сторінці самої групи, що дає змогу швидко порівнювати та контролювати фінансові показники як зі сторони працівника, так і зі сторони групи. Крім того, на сторінці групи передбачена можливість поставити або оновити ставку для викладача, що дозволяє коригувати оплату залежно від змін у навчальному процесі.

Процес встановлення ставки для викладача в групі відбувається автоматично, коли викладач додається до групи або безпосередньо до окремого заняття. При цьому система враховує низку параметрів, таких як розмір групи, тип групи, а також категорію викладача, що дозволяє максимально точно та справедливо розподіляти заробітну плату.

Для зручності користувача передбачена кнопка оплати, розташована у правій частині блоку, що містить інформацію про групи. Натиснувши цю кнопку, відкривається модальне вікно, у якому можна здійснити оплату нарахованих коштів, що значно спрощує та прискорює процес виплати заробітної плати викладачам (рис.3.30).

Оплата – Калякін Денис | Online School

Дата розрахунку: \* 23.05.2025

Дата оплати: \* 23.05.2025

Оплата: \* 1515

Як рахується "Остаточна оплата"  
 Остаточна оплата (1 515,00 грн) = Сума полів (1 515,00 грн) + Перевищення суми (0,00 грн) - Зайві кошти на рахунок (0,00 грн)

| Навчальна одиниця              | Нараховано   | Переплата                        | Заборгованість | Сума         |
|--------------------------------|--------------|----------------------------------|----------------|--------------|
| Тестова група (Женя)           | 0,00 грн     | 0,00 грн                         | 0,00 грн       | 0            |
| Тестова група (Макс)           | 1 515,00 грн | 0,00 грн                         | 1 515,00 грн   | 1515         |
| Загалом                        | 1 515,00 грн | 0,00 грн                         | 1 515,00 грн   | 1 515,00 грн |
| Перевіщення суми: 0,00 грн     |              | Зайві кошти на рахунок: 0,00 грн |                |              |
| Остаточна оплата: 1 515,00 грн |              |                                  |                |              |

Відмінити Зберегти

Рис.3.30. Модальне вікно для оплати заробітної плати.

У модальному вікні відображаються важливі параметри, що допомагають коректно провести операцію оплати заробітної плати викладачу. Зокрема, тут можна побачити дату розрахунку заробітної плати, тобто момент, коли була проведена калькуляція нарахованих сум, а також дату фактичної оплати, коли кошти будуть або вже були перераховані викладачу.

Крім цього, у вікні вказується сума коштів, яку ми хочемо внести на рахунок викладача. Для зручності і прозорості наведена таблиця, у якій представлені групи, до яких належить викладач, та детальна інформація про кожну з них. У цій таблиці можна побачити, скільки було нараховано за проведення занять у кожній конкретній групі, яка сума вже була переплачена компанією, а також скільки залишилось сплатити. Окремо виділена колонка, яка відображає перевищення суми - тут показуються зайві кошти, якщо під час оплати сума, яку ми намагаємося внести, більша за ту, що нарахувала система. У такому разі зайві гроші будуть зараховані на баланс викладача як надлишок, який можна буде врахувати при наступних виплатах. Це дає можливість уникнути помилок і зайвих виплат, одночасно зберігаючи гнучкість в управлінні фінансами.

Ще одна корисна функція полягає у тому, що можна вибірково оплатити лише певні групи. Для тих груп, по яких оплата не потрібна, можна просто вказати нуль у відповідній колонці «сума», що знаходиться праворуч від назви групи. Таким чином, система не включатиме ці групи до поточної операції оплати, що дає змогу керувати фінансами максимально точно і відповідно до потреб. Після завершення оплати, для підтвердження успішності виконаної операції, можна перейти до списку оплат, де відображаються всі проведені платежі, пов'язані з поточним викладачем. Це дозволяє відстежувати історію виплат, контролювати фінансові надходження та забезпечує повну прозорість у роботі з заробітною платою (рис. 3.31).

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         | 69   |

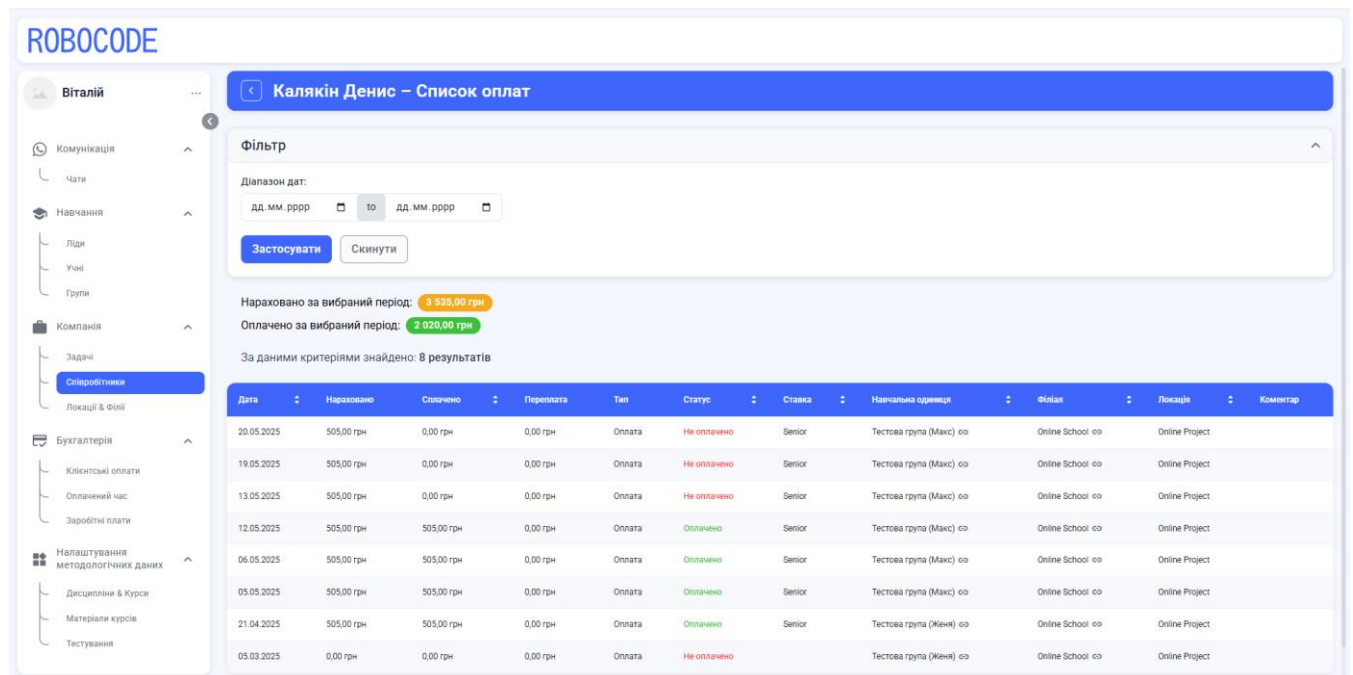


Рис. 3.31. Сторінка оплат викладача.

На цій сторінці передбачено фільтр з вибором діапазону дат, який дозволяє ефективно відслідковувати фінансові показники за певний проміжок часу. Завдяки цьому фільтру викладач або менеджер мають можливість переглянути загальну суму нарахованих коштів та фактично оплачених виплат за обраний період, що може бути корисно для аналізу роботи за місяць чи інший проміжок часу.

Окрім цього, на головній сторінці викладача розташовані кнопки, які дають змогу додавати додаткові виплати або робити вирахування з заробітної плати викладача.

Така функціональність дозволяє враховувати різні обставини, наприклад, можна створити доплату за участь у проведенні спеціальних заходів чи заходів поза основним графіком роботи, або навпаки - зробити вирахування у разі порушень, наприклад, запізнення на заняття. Це допомагає максимально точно вести облік виплат (рис. 3.32).

Доплата | Вирахування

×

Філіал:

Online School

▼

Дата:

дд.мм.рррр

📅

Дія:

Вирахування

▼

Сума оплати, грн:

150

Статус оплати:

Сплачено

▼

Коментар:

Тест

4/256

Відмінити

Зберегти

Рис. 3.32. Модальне вікно для додавання доплати або ж вирахування.

Коли ми додаємо доплату або здійснюємо вирахування із заробітної плати викладача, система надає можливість вказати кілька важливих параметрів, які забезпечують точний та повний облік цієї операції. Зокрема, потрібно ввести дату, до якої прив'язується дія, обрати відповідну філію, вказати точну суму, що підлягає доплаті або вирахуванню, визначити тип дії, тобто чи це додаткова виплата, чи віднімання коштів, а також обрати статус операції. Додатково можна залишити текстовий коментар, у якому пояснюється причина зміни у виплатах, що дозволяє вести прозорий внутрішній контроль.

Усі ці внесені дані не просто зберігаються в системі, а автоматично враховуються при розрахунку заробітної плати, що дає змогу уникати помилок і спрощує фінансову аналітику. Крім того, вони також потрапляють у загальну фінансову статистику по викладачу, що дозволяє мати повну картину змін, які відбувалися з виплатами в межах будь-якого періоду.

Перейдемо до функціональності з формування звітності. Розглянемо сторінку звіту про оплачений час (рис. 3.33).

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         | 71   |

Рис. 3.33. Сторінка звіту про оплачений час.

Для реалізації функціональності, пов'язаної з формуванням детальних звітів, був створений окремий report-сервіс. Це було зроблено через те, що сам процес побудови звіту досить ресурсоємний і потребує складної логіки обробки даних. Крім того, обсяг інформації, яку необхідно обробити, може бути суттєвим, особливо якщо обраний довгий період часу для звітності. Саме тому обробка звіту відбувається не миттєво. Після того як користувач заповнить необхідні поля фільтра і натисне кнопку «Застосувати», система ініціює процес побудови звіту. Зазвичай це займає приблизно 4–6 секунд, хоча точний час залежить від кількості інформації, що аналізується, та тривалості обраного звітного періоду.

Один із основних видів звітів, які формує система це звіт про оплачений час. Він подається у вигляді таблиці, яка містить важливу статистичну інформацію, зокрема загальну кількість проведених годин, кількість годин, що були оплачені, ставку оплати за годину, суму, яка була фактично виплачена, а також порівняння цих показників у розрізі кожного місяця. В кінці таблиці також автоматично генерується підсумкова частина, де зведено загальні показники за обраний період, що дає змогу швидко оцінити загальну картину без потреби вручну підраховувати значення.

Звіт про заробітну плату є важливою складовою контролю фінансових процесів. У цьому звіті автоматично формується таблиця, яка містить повну деталізацію усіх нарахувань заробітної плати за певний період. Зокрема, у таблиці

відображається загальна сума, яку система нарахувала викладачу відповідно до проведених ним занять, сума, яка була фактично виплачена, а також відображені всі групи, по яких здійснювались ці нарахування. Це дозволяє швидко перевірити, за яку саме діяльність було нараховано оплату, і порівняти її з фактичними виплатами, що були проведені.

Крім цього, у звіті подається супутня інформація, яка допомагає точніше зрозуміти структуру виплат. Наприклад, можна побачити, у якому саме місяці були зроблені нарахування, за які заняття й у яких філіях. Такий підхід дозволяє не лише вести облік нарахувань, але й оперативно реагувати на можливі розбіжності, виявляти помилки або уточнювати інформацію у разі виникнення спірних ситуацій. Звіт формується автоматично після вибору потрібного періоду, що значно спрощує процес контролю за нарахуваннями та виплатами заробітної плати (рис. 3.34).

Вітальї

Викладач

Чоловік

Нарахування

Ліди

Учні

Групи

Компанія

Задані

Співробітники

Локації & Філії

Бухгалтерія

Клієнтські оплати

Оплатений час

Заробітні плати

Нарахування методологічних дат

Дисципліни & Курси

Матеріальні курси

Тестування

Нарахування компаній

Співробітники

Локації & Філії

Ліди

Учні

Групи

Оплати

Інтеграція

API пошуку

Заробітні плати

Фільтр

Локація: Online Project

Філія: Не вибрано

Дата: 01.06.2025 до 09.06.2025

Викладач: Не вибрано

Створити звіт

Скинути

| Викладач / Філія                    | Група / Тип заняття                       | Статус / Інформація                        | Кількість занять | Оплатено | Параметри | Сума         |
|-------------------------------------|-------------------------------------------|--------------------------------------------|------------------|----------|-----------|--------------|
| Терех Андрій oo<br>Online School oo | Ср 20:00 Unity 2D oo                      | S-Middle teach. (425,00-535,00 грн)        | 1                | 0,00 грн | 0,00 грн  | 535,00 грн   |
|                                     | Ср 18:00 Unity 2D oo                      | S-Middle teach. (425,00-535,00 грн)        | 1                | 0,00 грн | 0,00 грн  | 535,00 грн   |
|                                     | Чт 18:00 Unity Beginner oo                | S-Middle teach. (425,00-535,00 грн)        | 1                | 0,00 грн | 0,00 грн  | 485,00 грн   |
|                                     | Чт 20:00 Unity 3D oo                      | S-Middle teach. Старий (560,00-560,00 грн) | 1                | 0,00 грн | 0,00 грн  | 560,00 грн   |
|                                     | Пн 18:00 Unity Beginner oo                | S-Middle teach. (425,00-535,00 грн)        | 1                | 0,00 грн | 0,00 грн  | 485,00 грн   |
|                                     | Пн 20:00 Unity Pro oo                     | S-Middle teach. Старий (560,00-560,00 грн) | 1                | 0,00 грн | 0,00 грн  | 560,00 грн   |
| Терех Віктор oo<br>Online School oo | Ср 11:00 Unity 3D oo                      | S-Middle teach. Старий (560,00-560,00 грн) | 1                | 0,00 грн | 0,00 грн  | 560,00 грн   |
|                                     | Вт 18:00 Unity Beginner Express Course oo | S-Middle teach. (425,00-535,00 грн)        | 1                | 0,00 грн | 0,00 грн  | 485,00 грн   |
|                                     |                                           |                                            | 8                | 0,00 грн | 0,00 грн  | 4 205,00 грн |
|                                     | Ср 21:00 Динаміка Волонтерів oo           | Middle teach. (350,00-350,00 грн)          | 1                | 0,00 грн | 0,00 грн  | 350,00 грн   |
|                                     | Ср 20:00 Динаміка Волонтерів oo           | Middle teach. (350,00-350,00 грн)          | 1                | 0,00 грн | 0,00 грн  | 350,00 грн   |
|                                     | Вт 20:00 Базис (Unity, Web, Python) oo    | Senior (505,00-415,00 грн)                 | 1                | 0,00 грн | 0,00 грн  | 560,00 грн   |
| Терех Андрій oo<br>Online School oo | Тестова група (Bme) oo                    | Senior (505,00-415,00 грн)                 | 2                | 0,00 грн | 0,00 грн  | 1 010,00 грн |
|                                     |                                           |                                            | 5                | 0,00 грн | 0,00 грн  | 2 270,00 грн |
|                                     | Вт 20:00 Unity 2D oo                      | Middle teach. (380,00-480,00 грн)          | 1                | 0,00 грн | 0,00 грн  | 420,00 грн   |
|                                     | Ср 20:00 Unity 3D oo                      | Middle teach. Старий (485,00-485,00 грн)   | 1                | 0,00 грн | 0,00 грн  | 485,00 грн   |
|                                     | Ср 11:00 Unity 2D oo                      | Middle teach. (380,00-480,00 грн)          | 1                | 0,00 грн | 0,00 грн  | 480,00 грн   |
|                                     | Ср 13:00 Unity Beginner oo                | Middle teach. (380,00-480,00 грн)          | 1                | 0,00 грн | 0,00 грн  | 480,00 грн   |

Рис. 3.34. Звіт про заробітні плати.

Також зверху над звітом можна побачити фільтр в якому ми можемо вказати локацію або філію по якій буде зформований звіт, діапазон дат та викладача для якого буде зформовано звіт. У разі якщо поле «Викладач» залишити порожнім, звіт буде сформовано по всіх викладачах локації або ж філії. Якщо поле «Філія» залишити порожнім, звіт буде зформовано по всіх філіях даної локації.

### 3.2 Сценарії використання системи користувачами різних ролей

У процесі створення CRM-системи для автоматизації фінансових процесів у школах особливу увагу приділено розмежуванню функціоналу між різними типами користувачів. Такий підхід дозволяє не лише підвищити зручність користування системою, а й забезпечити контроль доступу до окремих модулів, знизити ризики помилок або несанкціонованих дій з боку користувачів. Урахування ролей та прав доступу є критично важливим для систем, що працюють із фінансовою інформацією, де кожна дія має бути зафіксована, а повноваження - чітко визначені.

Система підтримує декілька типів ролей, кожна з яких відповідає реальним посадовим обов'язкам працівників школи або її клієнтів. Завдяки чіткому розмежуванню повноважень забезпечується логічна і функціональна ізоляція дій. Наприклад, адміністратор має можливість управляти користувачами, конфігурацією рахунків і модулями синхронізації, тоді як менеджер школи зосереджується переважно на операціях, пов'язаних із нарахуваннями та контролем оплат. Офісні працівники мають доступ до формування фінансової звітності та перегляду ключових показників, а викладачі взаємодіють переважно з інформацією про свої ставки, заняття та нараховану заробітну плату.

Такий підхід дає змогу реалізувати принцип найменших привілеїв - кожен користувач має доступ лише до тих функцій, які необхідні йому для виконання своїх обов'язків. Це не лише підвищує безпеку системи, а й дозволяє зробити інтерфейс більш зрозумілим і адаптованим до конкретної ролі. Усе це сприяє ефективній роботі школи та зменшує ймовірність людських помилок. У цьому підрозділі буде розглянуто основні сценарії використання системи представниками кожної з ролей, описано їх функціональні можливості, доступ до модулів, а також типові приклади дій, які виконуються в межах повсякденної роботи з CRM-системою.

Адміністратор є ключовою роллю в системі, яка забезпечує загальний контроль за всіма модулями CRM. Він має повний доступ до налаштування банківських рахунків, управління користувачами, а також відповідає за запуск

|      |      |                   |        |      |                                         |      |
|------|------|-------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А. |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.    |        |      |                                         |      |
| Змн. | Арк. | № докум.          | Підпис | Дата |                                         | 74   |

синхронізації з банківськими сервісами, зокрема API ПриватБанку. Адміністратор визначає параметри інтеграції, перевіряє статуси синхронізацій, керує правами доступу та ролями інших користувачів, а також здійснює аудит усіх фінансових дій через перегляд журналів і історії змін. Він може втручатися у всі процеси, від виправлення помилкових транзакцій до створення гарантів студентів. Ця роль найчастіше асоціюється з технічним керівником або головним адміністратором школи, який забезпечує стабільну і безпечну роботу всієї інформаційної системи.

Менеджер школи є безпосереднім виконавцем щоденних фінансових операцій. Його обов'язки охоплюють створення та редагування оплат на особові рахунки студентів, повернення коштів за невикористані заняття або залишки, а також контроль фактичного надходження коштів на рахунки. Менеджер відповідає за нарахування ставок викладачам, рознесення оплат по групах, перевірку статусів занять, а також взаємодію з системою у випадках часткової оплати чи змін у тарифах. Крім того, менеджер перевіряє правильність даних у системі, координує виплати зарплат викладачам, і часто виступає посередником між фінансовою частиною CRM та адміністрацією школи.

Викладач користується системою для перегляду свого розкладу, проведених занять та поточних нарахувань. Він може бачити інформацію про свої ставки, загальний обсяг викладання, а також суму заробітної плати, яку система автоматично розраховує на основі проведених занять. Також викладач має доступ до звітів про оплату часу, які дозволяють прозоро бачити, які саме заняття були оплачені, а які ще очікують оплати, що підвищує довіру до адміністрації та зменшує ризик непорозумінь.

Офісний працівник, як правило, виконує функції бухгалтера або фінансового аналітика. Його основне завдання - формування і перегляд фінансової звітності. У системі реалізовано кілька типів звітів, які охоплюють дані по оплатах клієнтів, оплаті часу викладачів, сумарному доходу, бонусах, залишках та ін. Працівник може фільтрувати ці звіти за філіями, періодами, групами або викладачами. Завдяки зручному інтерфейсу та аналітичним

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         | 75   |

можливостям, офісний персонал може швидко сформулювати звіт за будь-яким критерієм.

Студенти мають обмежений функціонал, спрямований переважно на освітню взаємодію. Це дозволяє зменшити навантаження на адміністрацію, адже студент самостійно отримує доступ до потрібної інформації без необхідності звертатися до менеджера або ж викладача.

Нижче наведено таблицю, яка узагальнює функціональні можливості кожної з ролей у системі.

Таблиця 3.1.

Доступні функції користувачам відповідно до ролей.

| Роль              | Доступні функції                                                                             |
|-------------------|----------------------------------------------------------------------------------------------|
| Адміністратор     | Повне управління системою, створення рахунків, керування користувачами, запуск синхронізації |
| Менеджер школи    | Створення та редагування оплат, повернення коштів, встановлення ставок, облік залишків       |
| Викладач          | Перегляд розкладу, перегляд зарплати, звіт про оплачений час                                 |
| Офісний працівник | Формування та перегляд фінансових звітів                                                     |
| Студент           | Немає доступу до фінансової функціональності                                                 |

### Висновки до третього розділу.

У цьому розділі було представлено опис інтерфейсу CRM-системи, яка забезпечує автоматизацію фінансових процесів у приватних школах. Надано текстовий та ілюстративний супровід, що демонструє основні елементи інтерфейсу, їх функціональне призначення та послідовність взаємодії користувача із системою. Описано методику роботи з ключовими модулями системи.

У ході тестування CRM-система яке відбувалось в реальному середовищі, була перевірена у максимально наближених до реальних умовах використання. В результаті тестування підтверджено стабільну та коректну роботу додатку, що свідчить про готовність продукту до впровадження в навчальних закладах.

## ВИСНОВКИ

Проведено аналіз існуючих CRM-систем для закладів освіти, зосереджених на фінансових процесах. Визначено ключові функціональні вимоги до системи, яку розробляється, а також сформовано архітектурне бачення рішення. Обґрунтовано вибір технологій, серед яких Spring Boot для бекенд-частини, база даних PostgreSQL, а також REST API для взаємодії між модулями. На основі аналізу було сформульовано мету, завдання, а також визначено підхід до реалізації функціоналу.

У другому розділі було розроблено архітектуру бази даних CRM-системи з урахуванням фінансових процесів, типових для приватних шкіл. Особливу увагу приділено автоматизації обробки оплат, інтеграції з банківськими рахунками, створенню гарантів користувачів та обліку транзакцій. Реалізовано мікросервіс авторизації, який забезпечує безпечну роботу з обліковими записами користувачів, дозволяє змінювати ролі, редагувати профіль і управляти доступом до функціоналу. В результаті реалізовано повноцінну серверну логіку та основні методи взаємодії з фінансовими модулями.

У третьому розділі представлено опис інтерфейсу CRM-системи та порядок роботи з додатком. Проведено тестування у середовищі, наближеному до реальних умов, під час якого система показала стабільну та надійну роботу. Детально описано основні вікна, інтерфейсні елементи, їх призначення та можливості. Інтерфейс реалізовано таким чином, щоб користувач міг інтуїтивно працювати з системою, здійснювати фінансові операції, переглядати звіти та взаємодіяти з іншими модулями програми.

У результаті виконання кваліфікаційної роботи було створено CRM-систему, яка автоматизує основні фінансові процеси приватних шкіл. Система відповідає вимогам, має зручний інтерфейс, масштабовану архітектуру та готова до практичного використання у реальному середовищі.

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         | 77   |

## ЛІТЕРАТУРНІ ДЖЕРЕЛА

1. Gutierrez F. Pro Spring Boot 3: An Authoritative Guide to Building Microservices, Web and Enterprise Applications, and Best Practices Using Spring Boot 3. – Apress, 2023. – 540 p.
2. King G., Bauer C. Java Persistence with Hibernate: Revised Edition. – Manning Publications, 2015. – 608 p.
3. Walls C. Spring in Action. 6th Edition. – Manning Publications, 2022. – 520 p.
4. Goetz B., Peierls T., Bloch J., Bowbeer J., Holmes D., Lea D. Java Concurrency in Practice. – Addison-Wesley, 2006. – 424 p.
5. Шарма Р. Реляційні бази даних. SQL та PostgreSQL. – К. : Наш Формат, 2020. – 412 с.
6. Бенкс Л. Docker для Java-розробника. – К. : Видавництво «Фабула», 2021. – 384 с.
7. Макдональд М. CI/CD для Java та Spring. – К. : Професіонал, 2022. – 350 с.
8. Apache Maven. Офіційна документація [Електронний ресурс]. – Режим доступу: <https://maven.apache.org>
9. Docker Documentation [Електронний ресурс]. – Режим доступу: <https://docs.docker.com>
10. Flyway by Redgate – Database Migrations Made Easy [Електронний ресурс]. – Режим доступу: <https://flywaydb.org>
11. Git. Офіційна документація [Електронний ресурс]. – Режим доступу: <https://git-scm.com/doc>
12. GitHub Docs [Електронний ресурс]. – Режим доступу: <https://docs.github.com>
13. Gradle Build Tool [Електронний ресурс]. – Режим доступу: <https://gradle.org>
14. IntelliJ IDEA. Офіційна документація [Електронний ресурс]. – Режим доступу: <https://www.jetbrains.com/idea>
15. Java SE Documentation. Oracle [Електронний ресурс]. – Режим доступу: <https://docs.oracle.com/javase>

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         | 78   |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         |      |

- 16.JDBC API. Oracle Docs [Електронний ресурс]. – Режим доступу:  
<https://docs.oracle.com/javase/8/docs/technotes/guides/jdbc>
- 17.JSON Web Tokens – Introduction [Електронний ресурс]. – Режим доступу:  
<https://jwt.io/introduction>
- 18.Liquibase Documentation [Електронний ресурс]. – Режим доступу:  
<https://www.liquibase.org>
- 19.MongoDB Documentation [Електронний ресурс]. – Режим доступу:  
<https://www.mongodb.com/docs>
- 20.MySQL 8.0 Reference Manual [Електронний ресурс]. – Режим доступу:  
<https://dev.mysql.com/doc>
- 21.Node.js Documentation [Електронний ресурс]. – Режим доступу:  
<https://nodejs.org/en/docs>
- 22.NPM Documentation [Електронний ресурс]. – Режим доступу:  
<https://docs.npmjs.com>
- 23.OpenAPI Specification – Swagger [Електронний ресурс]. – Режим доступу:  
<https://swagger.io/specification>
- 24.Postman API Platform Documentation [Електронний ресурс]. – Режим доступу: <https://learning.postman.com>
- 25.PostgreSQL Official Documentation [Електронний ресурс]. – Режим доступу:  
<https://www.postgresql.org/docs>
- 26.REST API Design Guidelines – Microsoft [Електронний ресурс]. – Режим доступу: <https://learn.microsoft.com/en-us/azure/architecture/best-practices/api-design>
- 27.Spring Boot Reference Documentation [Електронний ресурс]. – Режим доступу: <https://docs.spring.io/spring-boot/docs/current/reference/html>
- 28.Swagger UI Documentation [Електронний ресурс]. – Режим доступу:  
<https://swagger.io/tools/swagger-ui>
- 29.Spring Security Documentation [Електронний ресурс]. – Режим доступу:  
<https://docs.spring.io/spring-security/reference/index.html>

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         | 79   |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         |      |

- 30.Spring Data JPA Reference Documentation [Електронний ресурс]. – Режим доступу: <https://docs.spring.io/spring-data/jpa/docs/current/reference/html>
- 31.Hibernate ORM Documentation [Електронний ресурс]. – Режим доступу: <https://hibernate.org/orm/documentation>
- 32.Apache Tomcat Documentation [Електронний ресурс]. – Режим доступу: <https://tomcat.apache.org/tomcat-9.0-doc>
- 33.Lombok Project Documentation [Електронний ресурс]. – Режим доступу: <https://projectlombok.org>
- 34.MapStruct Documentation [Електронний ресурс]. – Режим доступу: <https://mapstruct.org/documentation>
- 35.Spring Cloud Documentation [Електронний ресурс]. – Режим доступу: <https://docs.spring.io/spring-cloud>
- 36.Grafana Documentation [Електронний ресурс]. – Режим доступу: <https://grafana.com/docs>

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         | 80   |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         |      |

# ДОДАТКИ

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         |      |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         | 81   |

## Допоміжний програмний код

```

@Slf4j
@Service
@RequiredArgsConstructor(onConstructor_ = {@Lazy})
public class SalaryServiceImpl implements SalaryService {

    private final WorkerMapper workerMapper;
    private final HistoryAuditService historyAuditService;
    private final SettingPaymentService settingPaymentService;
    private final TimeUnitMapper timeUnitMapper;
    private final TeacherGroupInfoService teacherGroupInfoService;
    private final GroupService groupService;
    private final TeacherLessonInfoService teacherLessonInfoService;
    private final WorkerService workerService;
    private final GroupMapper groupMapper;
    private final AdditionalChargeService additionalChargeService;
    private final AdditionalChargeMapper additionalChargeMapper;
    private final EmployeeCategoryMapper categoryMapper;

    @Override
    public List<GroupTeacherSalaryDto> fetchFinanceStatistics(Group group) {
        return group.getTeachersInfo().stream()
            .distinct()
            .map(this::fetchFinanceStatistics)
            .toList();
    }

    @Override
    public GroupTeacherSalaryDto fetchFinanceStatistics(TeacherGroupInfo teacherGroupInfo) {
        return fetchFinanceStatistics(teacherGroupInfo, null);
    }

    @Override
    public GroupTeacherSalaryDto fetchFinanceStatistics(TeacherGroupInfo teacherGroupInfo,
        LocalDate endDate) {
        return fetchFinanceStatistics(teacherGroupInfo, null, endDate);
    }

    @Override
    public List<GroupTeacherSalaryDto> fetchFinanceStatisticsBulk(List<Long> teacherIds,
        List<Long> branchOfficeIds, LocalDate dateMin, LocalDate dateMax) {
        var groupInfos = teacherGroupInfoService.getAllByTeachers(teacherIds, branchOfficeIds);
        var allLessons = groupInfos.stream()
            .flatMap(info -> info.getGroup().getLessons().stream())
            .filter(lesson -> LessonUtils.isLessonInRange(lesson, dateMin, dateMax))
            .distinct()
            .toList();
        var validLessons = filterValidLessons(allLessons);
        var teacherLessonsByTeacherId = getTeacherLessonsByTeacherId(validLessons);
    }

```

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         | 82   |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         |      |

```

        var groupLessonsByGroupId = getGroupLessonsByGroupId(validLessons);
        return groupInfos.stream()
            .map(info -> mapToSalaryDto(info, groupLessonsByGroupId, teacherLessonsByTeacherId))
            .toList();
    }

    private GroupTeacherSalaryDto mapToSalaryDto(TeacherGroupInfo info,
        Map<Long, List<Lesson>> groupLessonsByGroupId,
        Map<Long, List<TeacherLessonInfo>> teacherLessonsByTeacherId) {
        var teacher = info.getTeacher();
        var group = info.getGroup();
        var price = group.getPrice();
        var salaryRates = info.getTeacherSalaryRates();
        var salaryRateInfos = getSalaryRateInfos(salaryRates);
        var timeUnit = price != null ? price.getTimeParameter() : null;
        var groupLessons = groupLessonsByGroupId.getOrDefault(group.getId(), List.of());
        var lessonCount = groupLessons.stream()
            .filter(lesson -> LessonTeacherUtils.isTeacherPresentOnLesson(lesson, teacher))
            .filter(lesson -> !LessonTeacherUtils.hasTeacherReplacement(lesson, teacher))
            .count();
        var overpayment = calculateLessonOverpayment(teacher, group);
        var teacherLessons = teacherLessonsByTeacherId.getOrDefault(teacher.getId(), List.of());
        var teacherLessonInfosInGroup = teacherLessons.stream()
            .filter(infoItem -> groupLessons.contains(infoItem.getLesson()))
            .toList();
        var totalSalary = fetchCalculateSalary(teacherLessonInfosInGroup);
        var paidSalary = fetchPaidSalary(teacherLessonInfosInGroup);
        return buildGroupTeacherSalaryDto(group, teacher, salaryRateInfos, lessonCount, timeUnit,
            totalSalary, overpayment, paidSalary);
    }

    private static Map<Long, List<Lesson>> getGroupLessonsByGroupId(List<Lesson> validLessons)
    {
        return validLessons.stream()
            .filter(lesson -> Objects.nonNull(lesson.getGroup().getId()))
            .collect(Collectors.groupingBy(lesson -> lesson.getGroup().getId()));
    }

    private Map<Long, List<TeacherLessonInfo>> getTeacherLessonsByTeacherId(List<Lesson>
        validLessons) {
        return teacherLessonInfoService.getAllByLessonIn(validLessons).stream()
            .filter(info -> Objects.nonNull(info.getTeacher().getId()))
            .collect(Collectors.groupingBy(info -> info.getTeacher().getId()));
    }

    @Override
    public GroupTeacherSalaryDto fetchFinanceStatistics(TeacherGroupInfo teacherGroupInfo,
        LocalDate startDate, LocalDate endDate) {
        var teacher = teacherGroupInfo.getTeacher();
        var group = teacherGroupInfo.getGroup();
        var salaryRates = teacherGroupInfo.getTeacherSalaryRates();

```

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         | 83   |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         |      |

```

var salaryRateInfos = getSalaryRateInfos(salaryRates);
updateLatestSalaryRateInfo(salaryRates, salaryRateInfos);
var price = group.getPrice();
var timeUnit = Objects.nonNull(price) ? price.getTimeParameter() : null;
var groupLessons = fetchLessonsInRange(group.getLessons(), startDate, endDate);
var validLessons = filterValidLessons(groupLessons);
var lessonCount = validLessons.stream()
    .filter(lesson -> LessonTeacherUtils.isTeacherPresentOnLesson(lesson, teacher))
    .filter(lesson -> !LessonTeacherUtils.hasTeacherReplacement(lesson, teacher))
    .count();
var overpayment = calculateLessonOverpayment(teacher, group);
var teacherLessonInfos = teacherLessonInfoService.getAllByTeacherAndLessonIn(teacher,
validLessons);
var totalSalary = fetchCalculateSalary(teacherLessonInfos);
var paidSalary = fetchPaidSalary(teacherLessonInfos);
return buildGroupTeacherSalaryDto(group, teacher, salaryRateInfos, lessonCount, timeUnit,
totalSalary, overpayment, paidSalary);
}

private GroupTeacherSalaryDto buildGroupTeacherSalaryDto(Group group, Worker teacher,
List<SalaryRateBaseInfo> salaryRateInfos,
Long numberOfLesson, TimeUnit timeUnit, BigDecimal
totalSalary,
BigDecimal overpayment, BigDecimal paidSalary) {
return GroupTeacherSalaryDto.builder()
    .group(groupMapper.toBaseInfoWithoutSchedules(group))
    .teacher(workerMapper.toUserBaseInfoDto(teacher))
    .salaryRates(salaryRateInfos)
    .numberOfLesson(numberOfLesson)
    .hoursLessons(timeUnitMapper.toDto(timeUnit))
    .totalSalary(totalSalary)
    .overpayment(overpayment)
    .paidSalary(paidSalary)
    .build();
}

@Override
public BigDecimal fetchPaidSalary(List<TeacherLessonInfo> teacherLessonInfos) {
return teacherLessonInfos.stream()
    .map(TeacherLessonInfo::getPaidSalary)
    .reduce(BigDecimal.ZERO, BigDecimal::add);
}

@Override
public BigDecimal fetchCalculateSalary(List<TeacherLessonInfo> teacherLessonInfos) {
return teacherLessonInfos.stream()
    .map(TeacherLessonInfo::getCalculatedSalary)
    .reduce(BigDecimal.ZERO, BigDecimal::add);
}

private BigDecimal calculateLessonOverpayment(Worker teacher, Group group) {
var lessonInfos = teacherLessonInfoService.findAllByTeacherAndGroup(teacher, group);

```

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         | 84   |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         |      |

```

        return SalaryUtils.getOverpayment(lessonInfos);
    }

    private void updateLatestSalaryRateInfo(List<TeacherSalaryRate> salaryRates,
List<SalaryRateBaseInfo> salaryRateInfos) {
        if (salaryRateInfos.isEmpty() || salaryRates.isEmpty()) {
            return;
        }
        var latestSalaryRate = salaryRates.get(salaryRates.size() - 1);
        var latestSalaryRateInfo = salaryRateInfos.get(salaryRateInfos.size() - 1);
        var setting = historyAuditService.getByRevisionId(latestSalaryRate.getSettingRevisionId(),
SubcategorySetting.class);
        var latestRevisionId = historyAuditService.getRevisionId(setting.getId(),
SubcategorySetting.class);
        if (!Objects.equals(latestSalaryRate.getSettingRevisionId(), latestRevisionId)) {
            latestSalaryRateInfo.setActive(false);
        }
    }

    private List<Lesson> fetchLessonsInRange(List<Lesson> lessons, LocalDate startDate, LocalDate
endDate) {
        if (endDate == null) {
            return lessons;
        }
        return lessons.stream()
            .filter(lesson -> LessonUtils.isLessonInRange(lesson, startDate, endDate))
            .toList();
    }

    private List<Lesson> filterValidLessons(List<Lesson> lessons) {
        return lessons.stream()
            .filter(LessonUtils::isLessonCompleted)
            .filter(AttendanceUtils::hasStudents)
            .toList();
    }

    private List<SalaryRateBaseInfo> getSalaryRateInfos(List<TeacherSalaryRate> salaryRates) {
        return salaryRates.stream()
            .map(salaryRate -> {
                var startDate = salaryRate.getStartDate();
                var endDate = GroupUtils.getNextSalaryRateStartDate(salaryRates, salaryRate);
                return buildSalaryRateInfo(salaryRate, startDate, endDate);
            })
            .filter(Objects::nonNull)
            .collect(Collectors.toList());
    }

    private SalaryRateBaseInfo buildSalaryRateInfo(TeacherSalaryRate salaryRate, LocalDate
startDate, LocalDate endDate) {
        return
Optional.ofNullable(historyAuditService.getByRevisionId(salaryRate.getSettingRevisionId(),
SubcategorySetting.class))

```

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         | 85   |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         |      |

```

        .map(auditData -> {
            var subcategory = auditData.getSubcategory();
            var category = Objects.nonNull(subcategory) ? subcategory.getCategory() : null;
            return createSalaryRateBaseInfo(salaryRate.getId(), auditData.getSettingPayments(),
startDate, endDate, category);
        })
        .orElse(null);
    }

    private SalaryRateBaseInfo createSalaryRateBaseInfo(Long id, List<SettingPayment> payments,
LocalDate startDate, LocalDate endDate, EmployeeCategory category) {
        return SalaryRateBaseInfo.builder()
            .id(id)
            .minAmount(settingPaymentService.getAmount(payments, Comparator.naturalOrder()))
            .maxAmount(settingPaymentService.getAmount(payments, Comparator.reverseOrder()))
            .category(categoryMapper.toBaseInfo(category))
            .startDate(startDate)
            .endDate(endDate)
            .build();
    }

    @Override
    public SalaryRateBaseInfo buildSalaryRateInfo(TeacherSalaryRate salaryRate,
List<TeacherSalaryRate> salaryRates) {
        var startDate = salaryRate.getStartDate();
        var endDate = GroupUtils.getNextSalaryRateStartDate(salaryRates, salaryRate);
        return buildSalaryRateInfo(salaryRate, startDate, endDate);
    }

    @Override
    @Transactional
    public BigDecimal calculateLessonSalary(Lesson lesson, Worker teacher, List<TeacherSalaryRate>
salaryRates) {
        return calculateTotalSalary(List.of(lesson), teacher, salaryRates);
    }

    @Override
    @Transactional
    public void calculateSalaryForAllTeachers() {
        var teacherGroupInfos = teacherGroupInfoService.getAll();
        teacherGroupInfos
            .forEach(teacherGroupInfo -> {
                var lessons = teacherGroupInfo.getGroup().getLessons();
                var teacher = teacherGroupInfo.getTeacher();
                var salaryRates = teacherGroupInfo.getTeacherSalaryRates();
                calculateTotalSalary(lessons, teacher, salaryRates);
            });
    }

    @Override
    @Transactional(propagation = Propagation.REQUIRES_NEW)
    public BigDecimal calculateTotalSalary(List<Lesson> lessons, Worker teacher,

```

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         | 86   |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         |      |

```

List<TeacherSalaryRate> salaryRates) {
    var teacherLessonInfo = teacherLessonInfoService.getAllByTeacherAndLessonIn(teacher,
lessons);
    Map<TeacherSalaryRate, List<TeacherLessonInfo>> lessonsGroupedByRate =
getLessonsGroupedByRate(salaryRates, teacherLessonInfo);
    return lessonsGroupedByRate.entrySet().stream()
        .map(entry -> {
            var salaryRate = entry.getKey();
            var lessonInfos = entry.getValue();
            var group = lessonInfos.get(0).getLesson().getGroup();
            var settingPayments =
settingPaymentService.getSettingPaymentsFromSalaryRates(List.of(salaryRate), group.getType());
            var setting = historyAuditService.getByRevisionId(salaryRate.getSettingRevisionId(),
SubcategorySetting.class);
            var salaryType = setting.getSubcategory().getCategory().getSalaryType();
            var dependencyType =
setting.getSubcategory().getCategory().getSalaryDependencyType();
            return calculateSalaryByType(teacher, lessonInfos, settingPayments, salaryType,
salaryRate.getStartDate(),
                GroupUtils.getNextSalaryRateStartDate(salaryRates, salaryRate), dependencyType);
        })
        .reduce(BigDecimal.ZERO, BigDecimal::add);
}

private Map<TeacherSalaryRate, List<TeacherLessonInfo>>
getLessonsGroupedByRate(List<TeacherSalaryRate> salaryRates, List<TeacherLessonInfo>
teacherLessonInfo) {
    return teacherLessonInfo.stream()
        .map(info -> {
            var rate = findSalaryRateForLesson(info.getLesson(), salaryRates);
            return Objects.nonNull(rate) ? Map.entry(rate, info) : null;
        })
        .filter(Objects::nonNull)
        .collect(Collectors.groupingBy(Map.Entry::getKey,
            Collectors.mapping(Map.Entry::getValue, Collectors.toList())));
}

@Transactional
protected BigDecimal calculateSalaryByType(Worker teacher, List<TeacherLessonInfo>
lessonsInfo, List<SettingPayment> settingPayments,
            SalaryType salaryType, LocalDate startDate, LocalDate endDate,
            SalaryDependencyType dependencyType) {
    if (Objects.equals(SalaryType.PER_LESSON, salaryType)) {
        return calculateFixedPerLessonSalary(teacher, lessonsInfo, settingPayments, startDate,
endDate);
    }
    return calculateSalaryByDependencyType(teacher, lessonsInfo, startDate, endDate,
dependencyType, settingPayments);
}

@Transactional
protected BigDecimal calculateSalaryByDependencyType(Worker teacher,

```

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         | 87   |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         |      |

```

List<TeacherLessonInfo> lessonsInfo, LocalDate startDate, LocalDate endDate,
SalaryDependencyType dependencyType, List<SettingPayment>
settingPayments) {
    Function<TeacherLessonInfo, BigDecimal> paymentCalculator;
    if (Objects.equals(SalaryDependencyType.NUMBER_OF_STUDENTS_ON_LESSON,
dependencyType)) {
        paymentCalculator = lessonInfo -> calculateLessonPayment(lessonInfo.getLesson(),
settingPayments);
    } else if (Objects.equals(SalaryDependencyType.NUMBER_OF_STUDENTS_IN_GROUP,
dependencyType)) {
        paymentCalculator = lessonInfo ->
calculateGroupBasedLessonPayment(lessonInfo.getLesson(), settingPayments);
    } else {
        throw new
ValidationException(SalaryExceptionCodes.UNSUPPORTED_DEPENDENCY_TYPE,
dependencyType);
    }
    return calculateSalary(teacher, lessonsInfo, startDate, endDate, paymentCalculator);
}

@Transactional
protected BigDecimal calculateFixedPerLessonSalary(Worker teacher, List<TeacherLessonInfo>
lessonsInfo,
List<SettingPayment> settingPayments, LocalDate startDate,
LocalDate endDate) {
    var perLessonPayment = settingPaymentService.getFixedPerLessonPayment(settingPayments);
    return calculateSalary(teacher, lessonsInfo, startDate, endDate, lesson -> perLessonPayment);
}

private BigDecimal calculateLessonPayment(Lesson lesson, List<SettingPayment>
settingPayments) {
    var studentCount = AttendanceUtils.countStudentsOnLesson(lesson);
    return settingPaymentService.getRelevantSettingPayment(settingPayments, studentCount);
}

private BigDecimal calculateGroupBasedLessonPayment(Lesson lesson, List<SettingPayment>
settingPayments) {
    var group = lesson.getGroup();
    var groupSize = CollectionUtils.emptyIfNull(group.getStudentsInfo()).stream()
        .map(StudentGroupInfo::getStudentStatus)
        .filter(GroupStudentStatus.STUDIES::equals)
        .toList().size();
    return settingPaymentService.getRelevantSettingPayment(settingPayments, groupSize);
}

@Transactional
protected BigDecimal calculateSalary(Worker teacher, List<TeacherLessonInfo> lessonsInfo,
LocalDate startDate, LocalDate endDate,
Function<TeacherLessonInfo, BigDecimal> paymentCalculator) {
    var totalSalary = lessonsInfo.stream()
        .peek(info -> {
            var salary = LessonTeacherUtils.isLessonQualifiedForSalary(info.getLesson(), teacher,

```

|      |      |                   |        |      |                                         |      |
|------|------|-------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А. |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.    |        |      |                                         |      |
| Змн. | Арк. | № докум.          | Підпис | Дата |                                         | 88   |

```

startDate, endDate)
        ? paymentCalculator.apply(info)
        : BigDecimal.ZERO;
    info.setCalculatedSalary(salary);
})
.map(TeacherLessonInfo::getCalculatedSalary)
.reduce(BigDecimal.ZERO, BigDecimal::add);
teacherLessonInfoService.saveAll(lessonsInfo);
return totalSalary;
}

private TeacherSalaryRate findSalaryRateForLesson(Lesson lesson, List<TeacherSalaryRate>
salaryRates) {
    return salaryRates.stream()
        .filter(rate -> {
            var startDate = rate.getStartDate();
            var endDate = GroupUtils.getNextSalaryRateStartDate(salaryRates, rate);
            return LessonUtils.isLessonInRange(lesson, startDate, endDate);
        })
        .findFirst()
        .orElse(null);
}

@Override
public BigDecimal calculateTeacherSalaryDebt(Long teacherId) {
    var teacherLessonInfos = teacherLessonInfoService.findAllByTeacherAndPassed(teacherId);
    return teacherLessonInfos.stream()
        .map(this::fetchTeacherDebt)
        .reduce(BigDecimal.ZERO, BigDecimal::add);
}

private BigDecimal fetchTeacherDebt(TeacherLessonInfo teacherLessonInfo) {
    return teacherLessonInfo.getCalculatedSalary().subtract(teacherLessonInfo.getPaidSalary());
}

@Override
public GroupTeacherSalaryAndChargeDto getAllByTeacher(Long teacherId, LocalDate endDate,
Long branchOfficeId) {
    var teacher = workerService.getByIdAndRoleOrThrow(teacherId, Role.ROLE_TEACHER);
    var teacherSalaries = fetchTeacherGroupInfos(teacherId, branchOfficeId).stream()
        .map(teacherGroupInfo -> fetchFinanceStatistics(teacherGroupInfo, endDate))
        .filter(this::hasPositiveSalaryAndOverpayment)
        .sorted(Comparator
            .comparing((GroupTeacherSalaryDto dto) -> dto.getGroup().getType().ordinal())
            .thenComparing((GroupTeacherSalaryDto dto) -> dto.getGroup().getName(),
                Comparator.nullsLast(String::compareToIgnoreCase))
        )
        .toList();
    var teacherCharges = fetchAdditionalCharges(teacherId, branchOfficeId, endDate).stream()
        .map(additionalChargeMapper::toDto)
        .toList();
    return GroupTeacherSalaryAndChargeDto.builder()

```

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         | 89   |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         |      |

```

        .salaries(teacherSalaries)
        .additionalCharges(teacherCharges)
        .freeBalance(teacher.getFreeBalance())
        .build();
    }

    private boolean hasPositiveSalaryAndOverpayment(GroupTeacherSalaryDto
groupTeacherSalaryDto) {
        return Objects.nonNull(groupTeacherSalaryDto.getTotalSalary()) &&
groupTeacherSalaryDto.getTotalSalary().compareTo(BigDecimal.ZERO) != 0 ||
        Objects.nonNull(groupTeacherSalaryDto.getOverpayment()) &&
groupTeacherSalaryDto.getOverpayment().compareTo(BigDecimal.ZERO) != 0;
    }

    private List<TeacherGroupInfo> fetchTeacherGroupInfos(Long teacherId, Long branchOfficeId) {
        return Optional.ofNullable(branchOfficeId)
            .map(officeId -> teacherGroupInfoService.getAllByTeacher(teacherId, officeId))
            .orElseGet(() -> teacherGroupInfoService.getAllByTeacher(teacherId));
    }

    private List<AdditionalCharge> fetchAdditionalCharges(Long teacherId, Long branchOfficeId,
LocalDate endDate) {
        return Optional.ofNullable(branchOfficeId)
            .map(officeId ->
additionalChargeService.getAllByTeacherAndBranchOfficeAndNotPaid(teacherId, officeId,
endDate))
            .orElseGet(() -> additionalChargeService.getAllByTeacherAndNotPaid(teacherId,
endDate));
    }

    @Override
    @Transactional
    public GroupTeacherSalaryAndChargeDto applyTeacherSalaries(Long teacherId,
TeacherSalaryApplyDto teacherSalaryApplyDto) {
        var teacher = workerService.getByIdAndRoleOrThrow(teacherId, Role.ROLE_TEACHER);
        var additionalCharges = fetchAdditionalCharges(teacherSalaryApplyDto, teacher.getId());
        validateSalariesAndCharges(teacherSalaryApplyDto, additionalCharges);
        updateFreeBalance(teacher, teacherSalaryApplyDto, additionalCharges);
        var groupById = fetchGroupsById(teacherSalaryApplyDto);
        var resultSalaries = applyGroupSalaries(teacher, teacherSalaryApplyDto, groupById);
        var additionalChargeDtos = setAdditionalChargesPaid(additionalCharges);
        workerService.save(teacher);
        return buildSalaryAndChargeDto(resultSalaries, additionalChargeDtos, teacher);
    }

    private List<GroupTeacherSalaryDto> applyGroupSalaries(Worker teacher,
TeacherSalaryApplyDto dto, Map<Long, Group> groupById) {
        var operationDate = dto.getOperationDate();
        var applyDate = dto.getApplyDate();
        return dto.getGroupSalaries().stream()
            .sorted(Comparator
                .comparing((TeacherGroupSalaryApplyDto groupSalary) -> {

```

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         | 90   |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         |      |

```

        var group = groupId.get(groupId.getGroupId());
        return Objects.nonNull(group) ? group.getType().ordinal() : Integer.MAX_VALUE;
    })
    .thenComparing(groupId -> {
        var group = groupId.get(groupId.getGroupId());
        return Objects.nonNull(group) ? group.getName() : null;
    }, Comparator.nullsLast(String::compareToIgnoreCase))
    )
    .map(groupId -> applyTeacherSalary(teacher, groupId,
groupId.get(groupId.getGroupId()), operationDate, applyDate))
    .toList();
}

private void validateSalariesAndCharges(TeacherSalaryApplyDto dto, List<AdditionalCharge>
charges) {
    checkSalariesAndChargesCount(dto.getGroupSalaries(), charges);
    checkSalaryAmount(dto, charges);
}

private void updateFreeBalance(Worker teacher, TeacherSalaryApplyDto dto,
List<AdditionalCharge> charges) {
    var freeAmount = calculateFreeSalaryAmount(dto, charges);
    teacher.setFreeBalance(freeAmount);
}

private Map<Long, Group> fetchGroupsById(TeacherSalaryApplyDto dto) {
    var groupIds = dto.getGroupSalaries().stream()
        .map(TeacherGroupSalaryApplyDto::getGroupId)
        .toList();
    return groupIdService.findAllByIds(groupIds).stream()
        .collect(Collectors.toMap(Group::getId, Function.identity()));
}

private GroupTeacherSalaryAndChargeDto
buildSalaryAndChargeDto(List<GroupTeacherSalaryDto> resultSalaries,
                        List<AdditionalChargeDto> chargeDtos, Worker teacher) {
    return GroupTeacherSalaryAndChargeDto.builder()
        .salaries(resultSalaries)
        .additionalCharges(chargeDtos)
        .freeBalance(teacher.getFreeBalance())
        .build();
}

private BigDecimal calculateFreeSalaryAmount(TeacherSalaryApplyDto teacherSalaryApplyDto,
List<AdditionalCharge> additionalCharges) {
    var totalAmountForGroups = teacherSalaryApplyDto.getGroupSalaries().stream()
        .map(TeacherGroupSalaryApplyDto::getAmount)
        .reduce(BigDecimal.ZERO, BigDecimal::add);
    var totalAmountAdditionalCharges = additionalCharges.stream()
        .map(AdditionalCharge::getAmount)
        .reduce(BigDecimal.ZERO, BigDecimal::add);
    var totalAmount = totalAmountForGroups.add(totalAmountAdditionalCharges);
}

```

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         | 91   |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         |      |

```

        return teacherSalaryApplyDto.getTotalAmount().subtract(totalAmount).max(BigDecimal.ZERO);
    }

    private List<AdditionalCharge> fetchAdditionalCharges(TeacherSalaryApplyDto
teacherSalaryApplyDto, Long teacherId) {
        var chargeIds = teacherSalaryApplyDto.getPaidAdditionalChargeIds();
        return additionalChargeService.getAllByIdsAndTeacherAndNotPaid(chargeIds, teacherId);
    }

    private void checkSalariesAndChargesCount(List<TeacherGroupSalaryApplyDto> groupSalaries,
List<AdditionalCharge> additionalCharges) {
        if (CollectionUtils.size(groupSalaries) + CollectionUtils.size(additionalCharges) == 0) {
            throw new
ValidationException(TeacherLessonSalaryExceptionCodes.AMOUNTS_ARE_EMPTY);
        }
    }

    private void checkSalaryAmount(TeacherSalaryApplyDto teacherSalaryApplyDto,
List<AdditionalCharge> additionalCharges) {
        var totalAmount = teacherSalaryApplyDto.getTotalAmount();
        var salaryAmount =
CollectionUtils.emptyIfNull(teacherSalaryApplyDto.getGroupSalaries()).stream()
            .map(TeacherGroupSalaryApplyDto::getAmount)
            .reduce(BigDecimal.ZERO, BigDecimal::add);
        var chargeAmount = CollectionUtils.emptyIfNull(additionalCharges).stream()
            .map(charge -> AdditionalChargeType.ADD.equals(charge.getType()) ?
                charge.getAmount() : charge.getAmount().negate())
            .reduce(BigDecimal.ZERO, BigDecimal::add);
        var actualAmount = salaryAmount.add(chargeAmount);
        if (actualAmount.compareTo(BigDecimal.ZERO) < 0) {
            throw new
ValidationException(TeacherLessonSalaryExceptionCodes.ACTUAL_AMOUNT_IS_NEGATIVE,
                actualAmount.setScale(AMOUNT_SCALE, AMOUNT_ROUNDING).doubleValue());
        }
        if (actualAmount.compareTo(totalAmount) > 0) {
            throw new
ValidationException(TeacherLessonSalaryExceptionCodes.TOTAL_AMOUNT_IS_LESS_THAN_SAL
ARY_SUM,
                totalAmount.setScale(AMOUNT_SCALE, AMOUNT_ROUNDING).doubleValue(),
                salaryAmount.setScale(AMOUNT_SCALE, AMOUNT_ROUNDING).doubleValue());
        }
    }

    private GroupTeacherSalaryDto applyTeacherSalary(Worker teacher,
TeacherGroupSalaryApplyDto groupSalary,
                Group group, LocalDate operationDate, LocalDate applyDate) {
        var teacherGroupInfo = GroupUtils.fetchTeacherGroupInfoOrThrow(group, teacher.getId());
        var teacherLessonInfos = teacherLessonInfoService.getAllByGroupAndTeacher(group,
teacher).stream()
            .filter(info -> LessonUtils.isLessonInRange(info.getLesson(), null, operationDate))
            .toList();
        var groupOverpayment = teacherLessonInfos.stream()

```

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         | 92   |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         |      |

```

        .map(TeacherLessonInfo::getOverpayment)
        .reduce(BigDecimal.ZERO, BigDecimal::add);
var amount = groupSalary.getAmount().add(groupOverpayment);
if (AmountCalcUtils.equalsZero(amount)) {
    return fetchFinanceStatistics(teacherGroupInfo);
}
applyTeacherSalary(teacherLessonInfos, teacher, amount, applyDate);
return fetchFinanceStatistics(teacherGroupInfo);
}

private void applyTeacherSalary(List<TeacherLessonInfo> teacherLessonInfos, Worker teacher,
BigDecimal amount, LocalDate applyDate) {
    List<TeacherLessonInfo> updatedTeacherLessonInfos = new ArrayList<>();
    for (var teacherLessonInfo : teacherLessonInfos) {
        amount = applyTeacherSalary(teacherLessonInfo, amount, applyDate,
updatedTeacherLessonInfos);
        if (AmountCalcUtils.equalsZero(amount)) {
            break;
        }
    }
    teacher.setFreeBalance(teacher.getFreeBalance().add(amount));
    teacherLessonInfoService.saveAll(updatedTeacherLessonInfos);
}

private BigDecimal applyTeacherSalary(TeacherLessonInfo teacherLessonInfo, BigDecimal
amount,
        LocalDate applyDate, List<TeacherLessonInfo>
updatedTeacherLessonInfos) {
    var salary = teacherLessonInfo.getCalculatedSalary();
    return applyTeacherSalaryAmount(teacherLessonInfo, salary, amount, applyDate,
updatedTeacherLessonInfos);
}

private BigDecimal applyTeacherSalaryAmount(TeacherLessonInfo teacherLessonInfo,
BigDecimal salary, BigDecimal amount,
        LocalDate applyDate, List<TeacherLessonInfo>
updatedTeacherLessonInfos) {
    var salaryAmount = teacherLessonInfo.getPaidSalary();
    var applyAmount = salary.subtract(salaryAmount).min(amount);
    if (AmountCalcUtils.equalsZero(applyAmount)) {
        return amount;
    }
    var newSalaryAmount = salaryAmount.add(applyAmount);
    var teacherLessonSalary = buildTeacherLessonSalary(teacherLessonInfo, applyAmount,
applyDate);
    teacherLessonInfo.setPaidSalary(newSalaryAmount);
    teacherLessonInfo.getSalaries().add(teacherLessonSalary);
    updatedTeacherLessonInfos.add(teacherLessonInfo);
    var overpayment = teacherLessonInfo.getOverpayment();
    if (overpayment.compareTo(BigDecimal.ZERO) > 0) {
        teacherLessonInfo.setOverpayment(BigDecimal.ZERO);
    }
}

```

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         | 93   |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         |      |

```

        return amount.subtract(applyAmount);
    }

    private TeacherLessonSalary buildTeacherLessonSalary(TeacherLessonInfo teacherLessonInfo,
        BigDecimal amount,
                        LocalDate applyDate) {
        return TeacherLessonSalary.builder()
            .teacherLessonInfo(teacherLessonInfo)
            .applyDate(applyDate)
            .amount(amount)
            .build();
    }

    private List<AdditionalChargeDto> setAdditionalChargesPaid(List<AdditionalCharge>
        additionalCharges) {
        return additionalChargeService.setPaid(additionalCharges).stream()
            .map(additionalChargeMapper::toDto)
            .toList();
    }
}

@Service
@RequiredArgsConstructor
public class TeacherFinanceServiceImpl implements TeacherFinanceService {

    private final TeacherLessonInfoService teacherLessonInfoService;
    private final WorkerService workerService;
    private final AdditionalChargeService additionalChargeService;
    private final AdditionalChargeMapper additionalChargeMapper;
    private final TeacherLessonSalaryService teacherLessonSalaryService;
    private final SalaryService salaryService;
    private final GroupService groupService;
    private final GroupMapper groupMapper;
    private final BranchOfficeMapper branchOfficeMapper;

    @Override
    public TeacherSalaryItemsBaseInfoDto getAllTeacherSalaryItemsCalendar(Long teacherId,
        LocalDate date,
                        TeacherSalaryItemDateDirection direction,
                        Integer pageSize) {
        validateItemsFetchPossibility(date, direction);
        var teacher = workerService.getByIdAndRoleOrThrow(teacherId, Role.ROLE_TEACHER);
        var dates = fetchDates(teacherId, date, direction, pageSize);
        var teacherLessonInfos = teacherLessonInfoService.findAllByTeacherAndDates(teacher.getId(),
        dates);
        var additionalCharges =
        additionalChargeService.getAllByTeacherAndApplyDates(teacher.getId(), dates);
        return buildSalaryItemsCalendar(teacherLessonInfos, additionalCharges, teacher.getId(), dates);
    }

    private void validateItemsFetchPossibility(LocalDate date, TeacherSalaryItemDateDirection
        direction) {

```

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         | 94   |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         |      |

```

        if ((date == null) != (direction == null)) {
            throw new ValidationException(LessonExceptionCodes.SALARY_ITEMS_INVALID_RANGE,
date, direction);
        }
    }

    private List<LocalDate> fetchDates(Long teacherId, LocalDate date,
TeacherSalaryItemDateDirection direction,
        Integer pageSize) {
        if (TeacherSalaryItemDateDirection.BEFORE.equals(direction)) {
            return teacherLessonInfoService.getAllLessonDatesBeforeDate(teacherId, date, pageSize);
        }
        if (TeacherSalaryItemDateDirection.AFTER.equals(direction)) {
            return teacherLessonInfoService.getAllLessonDatesAfterDate(teacherId, date, pageSize);
        }
        return teacherLessonInfoService.getAllLessonDatesAroundNow(teacherId, pageSize);
    }

    private TeacherSalaryItemsBaseInfoDto buildSalaryItemsCalendar(List<TeacherLessonInfo>
teacherLessonInfos,
        List<AdditionalCharge> additionalCharges,
        Long teacherId, List<LocalDate> dates) {
        var lessonSalaryItems = buildLessonSalaryItems(teacherLessonInfos, additionalCharges);
        var chargeSalaryItems = buildChargeSalaryItems(additionalCharges);
        var payments = unionSalaryItemLists(lessonSalaryItems, chargeSalaryItems);
        var debtAmount = calculateDebtAmount(teacherId);
        var start = dates.stream().min(Comparator.naturalOrder()).orElse(null);
        var end = dates.stream().max(Comparator.naturalOrder()).orElse(null);
        var datesBefore = teacherLessonInfoService.countAllByTeacherAndBeforeDate(teacherId, start);
        var datesAfter = teacherLessonInfoService.countAllByTeacherAndAfterDate(teacherId, end);
        var overpayment = calculateOverpaymentForTeacher(teacherId);
        return TeacherSalaryItemsBaseInfoDto.builder()
            .salaryItemsBefore(datesBefore)
            .salaryItemsAfter(datesAfter)
            .debtAmount(debtAmount)
            .salaryItems(payments)
            .overpayment(overpayment)
            .build();
    }

    private BigDecimal calculateOverpaymentForTeacher(Long teacherId) {
        var teacher = workerService.getByIdAndRoleOrThrow(teacherId, Role.ROLE_TEACHER);
        var teacherLessonInfos = teacherLessonInfoService.getAllByTeacher(teacher);
        return SalaryUtils.getOverpayment(teacherLessonInfos);
    }

    private List<TeacherSalaryItemBaseInfoDto> buildLessonSalaryItems(List<TeacherLessonInfo>
teacherLessonInfos,
        List<AdditionalCharge> additionalCharges) {
        return teacherLessonInfos.stream()
            .map(teacherLessonInfo -> buildLessonSalaryItem(teacherLessonInfo, additionalCharges))
            .collect(Collectors.groupingBy(TeacherSalaryItemBaseInfoDto::getDate))
    }

```

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         | 95   |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         |      |

```

        .entrySet()
        .stream()
        .map(entry -> mergeLessonSalaryItems(entry.getKey(), entry.getValue()))
        .toList();
    }

    private TeacherSalaryItemBaseInfoDto buildLessonSalaryItem(TeacherLessonInfo
teacherLessonInfo,
                                List<AdditionalCharge> additionalCharges) {
        var date = teacherLessonInfo.getLesson().getStart().toLocalDate();
        var timeAmount = Optional.ofNullable(teacherLessonInfo.getLesson().getGroup().getPrice())
            .map(price -> price.getTimeParameter().getMinutes())
            .orElse(0);
        var charges = additionalCharges.stream()
            .filter(charge -> date.equals(charge.getApplyDate()))
            .toList();
        additionalCharges.removeAll(charges);
        var chargeDtos = additionalChargeMapper.toDtos(charges);
        var salaryStatus = fetchLessonAndChargeStatuses(teacherLessonInfo, charges);
        return TeacherSalaryItemBaseInfoDto.builder()
            .date(date)
            .timeAmount(timeAmount)
            .additionalCharges(chargeDtos)
            .salaryStatus(salaryStatus)
            .build();
    }

    private EmployeeSalaryStatus fetchLessonAndChargeStatuses(TeacherLessonInfo
teacherLessonInfo,
                                List<AdditionalCharge> charges) {
        var lessonSalaryStatus = LessonTeacherUtils.fetchLessonSalaryStatus(teacherLessonInfo);
        if (charges.isEmpty()) {
            return lessonSalaryStatus;
        }
        var chargesPaidStatus = fetchChargesPaidStatus(charges);
        if (lessonSalaryStatus.equals(chargesPaidStatus)) {
            return lessonSalaryStatus;
        }
        if (EmployeeSalaryStatus.NOT_PAID_SKIP.equals(lessonSalaryStatus) ||
            EmployeeSalaryStatus.NOT_PASSED_LESSON.equals(lessonSalaryStatus)) {
            return chargesPaidStatus;
        }
        return EmployeeSalaryStatus.PARTLY_PAID;
    }

    private TeacherSalaryItemBaseInfoDto mergeLessonSalaryItems(LocalDate date,
List<TeacherSalaryItemBaseInfoDto> payments) {
        var timeAmount = calculateTimeAmount(payments);
        var charges = mergeCharges(payments);
        var salaryStatus = mergeSalaryStatuses(payments);
        return TeacherSalaryItemBaseInfoDto.builder()
            .date(date)

```

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         | 96   |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         |      |

```

        .timeAmount(timeAmount)
        .additionalCharges(charges)
        .salaryStatus(salaryStatus)
        .build();
    }

    private Integer calculateTimeAmount(List<TeacherSalaryItemBaseInfoDto> teacherSalaryItems) {
        return teacherSalaryItems.stream()
            .mapToInt(TeacherSalaryItemBaseInfoDto::getTimeAmount)
            .sum();
    }

    private List<AdditionalChargeDto> mergeCharges(List<TeacherSalaryItemBaseInfoDto>
payments) {
        return payments.stream()
            .map(TeacherSalaryItemBaseInfoDto::getAdditionalCharges)
            .flatMap(Collection::stream)
            .toList();
    }

    private EmployeeSalaryStatus mergeSalaryStatuses(List<TeacherSalaryItemBaseInfoDto>
payments) {
        var statusesCount = payments.stream()
            .collect(Collectors.groupingBy(TeacherSalaryItemBaseInfoDto::getSalaryStatus,
Collectors.counting()));
        if (statusesCount.getDefault(EmployeeSalaryStatus.NOT_PASSED_LESSON, 0L) > 0) {
            return EmployeeSalaryStatus.NOT_PASSED_LESSON;
        }
        var notPaidSkipCount = statusesCount.getDefault(EmployeeSalaryStatus.NOT_PAID_SKIP,
0L);
        var paymentsCount = payments.size();
        if (notPaidSkipCount == paymentsCount) {
            return EmployeeSalaryStatus.NOT_PAID_SKIP;
        }
        var paidCount = statusesCount.getDefault(EmployeeSalaryStatus.FULLY_PAID, 0L) +
notPaidSkipCount;
        if (paidCount == paymentsCount) {
            return EmployeeSalaryStatus.FULLY_PAID;
        }
        var notPaidCount = statusesCount.getDefault(EmployeeSalaryStatus.NOT_PAID, 0L) +
notPaidSkipCount;
        if (notPaidCount == paymentsCount) {
            return EmployeeSalaryStatus.NOT_PAID;
        }
        return EmployeeSalaryStatus.PARTLY_PAID;
    }

    private List<TeacherSalaryItemBaseInfoDto> buildChargeSalaryItems(List<AdditionalCharge>
additionalCharges) {
        return additionalCharges.stream()
            .collect(Collectors.groupingBy(AdditionalCharge::getApplyDate))
            .entrySet()

```

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         | 97   |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         |      |

```

        .stream()
        .map(entry -> buildChargeSalaryItem(entry.getKey(), entry.getValue()))
        .sorted(Comparator.comparing(TeacherSalaryItemBaseInfoDto::getDate).reversed())
        .toList();
    }

    private TeacherSalaryItemBaseInfoDto buildChargeSalaryItem(LocalDate date,
                                                                List<AdditionalCharge> additionalCharges) {
        var additionalChargeDtos = additionalChargeMapper.toDtos(additionalCharges);
        return TeacherSalaryItemBaseInfoDto.builder()
            .date(date)
            .additionalCharges(additionalChargeDtos)
            .salaryStatus(fetchChargesPaidStatus(additionalCharges))
            .build();
    }

    private EmployeeSalaryStatus fetchChargesPaidStatus(List<AdditionalCharge> additionalCharges)
    {
        var statusesCount = additionalCharges.stream()
            .collect(Collectors.groupingBy(AdditionalCharge::getStatus, Collectors.counting()));
        var chargesCount = additionalCharges.size();
        if (statusesCount.getOrDefault(AdditionalChargeStatus.PAID, 0L) == chargesCount) {
            return EmployeeSalaryStatus.FULLY_PAID;
        }
        if (statusesCount.getOrDefault(AdditionalChargeStatus.NOT_PAID, 0L) == chargesCount) {
            return EmployeeSalaryStatus.NOT_PAID;
        }
        return EmployeeSalaryStatus.PARTLY_PAID;
    }

    private List<TeacherSalaryItemBaseInfoDto>
    unionSalaryItemLists(List<TeacherSalaryItemBaseInfoDto> lessonSalaryItems,
                        List<TeacherSalaryItemBaseInfoDto> chargeSalaryItems) {
        return ListUtils.union(lessonSalaryItems, chargeSalaryItems).stream()
            .sorted(Comparator.comparing(TeacherSalaryItemBaseInfoDto::getDate))
            .toList();
    }

    private BigDecimal calculateDebtAmount(Long teacherId) {
        var salaryDebtAmount = salaryService.calculateTeacherSalaryDebt(teacherId);
        var chargesDebtAmount =
        additionalChargeService.calculateAmountSumByTeacherAndNotPaid(teacherId);
        return salaryDebtAmount.add(chargesDebtAmount);
    }

    @Override
    public TeacherSalaryItemsDto getAllTeacherSalaryItems(Long teacherId, DateFilteredDto
    dateFilteredDto) {
        var dateRange = normalizeDateRange(dateFilteredDto);
        var dateMin = dateRange.a;
        var dateMax = dateRange.b;
        var salaryItemPage = teacherLessonSalaryService.getAllByTeacher(teacherId, dateMin, dateMax,

```

|      |      |                   |        |      |                                         |      |
|------|------|-------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А. |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.    |        |      |                                         | 98   |
| Змн. | Арк. | № докум.          | Підпис | Дата |                                         |      |

```

dateFilteredDto);
    var projections = salaryItemPage.getContent();
    var teacherLessonInfoIds =
projections.stream().map(TeacherSalaryItemProjection::getTeacherLessonInfoId).toList();
    var teacherLessonInfos = teacherLessonInfoService.findAllByIds(teacherLessonInfoIds);
    var salaryItems = salaryItemPage.getContent().stream()
        .map(projection -> projectionToSalaryItem(projection, teacherId, teacherLessonInfos))
        .toList();
    var filteredLessonInfos =
teacherLessonInfoService.findAllByTeacherIdAndDateRange(teacherId, dateMin, dateMax);
    var calculatedSalary = salaryService.fetchCalculateSalary(filteredLessonInfos);
    var paidSalary = salaryService.fetchPaidSalary(filteredLessonInfos);
    return TeacherSalaryItemsDto.builder()
        .salaryItems(salaryItems)
        .currentPage(salaryItemPage.getNumber() + 1)
        .totalPages(salaryItemPage.getTotalPages())
        .totalMatches(salaryItemPage.getTotalElements())
        .calculatedSalary(calculatedSalary)
        .paidSalary(paidSalary)
        .build();
}

```

```

public Pair<LocalDateTime, LocalDateTime> normalizeDateRange(DateFilteredDto
dateFilteredDto) {
    var dateMin = dateFilteredDto.getDateMin();
    var dateMax = dateFilteredDto.getDateMax();
    LocalDateTime minDateTime = null;
    if (Objects.nonNull(dateMin)) {
        minDateTime = dateMin.atStartOfDay();
    }
    if (Objects.isNull(dateMax) || dateMax.isAfter(LocalDate.now())) {
        dateMax = LocalDate.now();
        dateFilteredDto.setDateMax(dateMax);
    }
    var maxDateTime = dateMax.equals(LocalDate.now())
        ? LocalDateTime.of(dateMax, LocalTime.now())
        : dateMax.atTime(LocalTime.MAX);
    return new Pair<>(minDateTime, maxDateTime);
}

```

```

private TeacherSalaryItemDto projectionToSalaryItem(TeacherSalaryItemProjection projection,
Long teacherId,
List<TeacherLessonInfo> teacherLessonInfos) {
    var group = groupService.findById(projection.getGroupId()).orElse(null);
    var teacherLessonInfo = teacherLessonInfos.stream()
        .filter(info -> Objects.equals(info.getId(), projection.getTeacherLessonInfoId()))
        .findFirst()
        .orElse(null);
    var salaryRateDto = buildSalaryRate(group, projection.getLessonDateTime(), teacherId);
    var branchOfficeDto = branchOfficeMapper.projectionToBranchOfficeBaseInfoDto(projection);
    var salaryStatus = fetchSalaryStatus(projection, teacherLessonInfo);
    var totalSalary = fetchTotalSalary(projection, teacherLessonInfo);
}

```

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         | 99   |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         |      |

```

var overpayment = projection.getOverpayAmount();
return TeacherSalaryItemDto.builder()
    .applyDate(projection.getApplyDate())
    .group(groupMapper.toBaseInfo(group))
    .branchOffice(branchOfficeDto)
    .description(projection.getDescription())
    .amount(projection.getAmount())
    .totalSalary(totalSalary)
    .type(projection.getType())
    .salaryRate(salaryRateDto)
    .salaryStatus(salaryStatus)
    .overpayment(overpayment)
    .build();
}

private SalaryRateBaseInfo buildSalaryRate(Group group, LocalDateTime lessonDateTime, Long
teacherId) {
    if (group == null) {
        return null;
    }
    var teacherGroupInfo = GroupUtils.fetchTeacherGroupInfoOrThrow(group, teacherId);
    var salaryRates = teacherGroupInfo.getTeacherSalaryRates();
    var salaryRate = fetchSalaryRate(salaryRates, lessonDateTime);
    if (salaryRate == null) {
        return null;
    }
    return salaryService.buildSalaryRateInfo(salaryRate, salaryRates);
}

private TeacherSalaryRate fetchSalaryRate(List<TeacherSalaryRate> salaryRates, LocalDateTime
lessonDateTime) {
    var lessonDate = lessonDateTime.toLocalDate();
    return salaryRates.stream()
        .filter(salaryRate -> {
            var startDate = salaryRate.getStartDate();
            var endDate = GroupUtils.getNextSalaryRateStartDate(salaryRates, salaryRate);
            return DateTimeUtils.isBetween(lessonDate, startDate, endDate);
        })
        .findAny()
        .orElse(null);
}

private EmployeeSalaryStatus fetchSalaryStatus(TeacherSalaryItemProjection projection,
TeacherLessonInfo teacherLessonInfo) {
    if (!TeacherPaymentType.SALARY.equals(projection.getType())) {
        return projection.getSalaryStatus();
    }
    return LessonTeacherUtils.fetchLessonSalaryStatus(teacherLessonInfo);
}

private BigDecimal fetchTotalSalary(TeacherSalaryItemProjection projection, TeacherLessonInfo
teacherLessonInfo) {

```

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         | 100  |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         |      |

```

        if (!TeacherPaymentType.SALARY.equals(projection.getType())) {
            return projection.getAmount();
        }
        var lesson = teacherLessonInfo.getLesson();
        var teacher = teacherLessonInfo.getTeacher();
        var teacherLessonInfos = teacherLessonInfoService.getByTeacherAndLesson(teacher, lesson);
        return teacherLessonInfos.getCalculatedSalary();
    } @Slf4j

    public abstract class IntegrationBankService {

        protected abstract RootClient getRootClient();

        protected abstract BankIntegrationRepository getBankIntegrationRepository();

        protected abstract SynchronizationAsyncService getSynchronizationAsyncService();

        public abstract String getBankCode();

        public abstract ResponseEntity<?> executeConnectionTest(Map<String, String> params);

        public abstract void executeSynchronize(Map<String, String> stringStringMap,
            SynchronizeRequestDto synchronizeRequestDto, BankAccountBaseInfoDto bankAccount);

        public abstract Map<String, String> buildParams(BankIntegration bankIntegration,
            BankAccountBaseInfoDto bankAccount);

        public IntegrationStatus testConnection(BankAccountBaseInfoDto bankAccount, BankIntegration
            bankIntegration) {
            var params = buildParams(bankIntegration, bankAccount);
            var integrationUnsupported = IntegrationBankUtils.getIntegrationStatus(bankAccount,
            bankIntegration);
            if (Objects.nonNull(integrationUnsupported)) {
                return integrationUnsupported;
            }
            try {
                var statusCode = executeConnectionTest(params).getStatusCode();
                return IntegrationBankUtils.getIntegrationStatusForCode(statusCode);
            } catch (IllegalArgumentException e) {
                throw new IntegrationException(BankExceptionCodes.CONNECTION_FAILED);
            }
        }

        public void updateIntegrationStatus(Long bankAccountId) {
            IntegrationStatus status;
            var bankAccount = getRootClient().fetchBankAccount(bankAccountId);
            var bankIntegration =
            getBankIntegrationRepository().getBankFieldsValueByBankAccountId(bankAccountId);
            if (!isBankSystem(bankAccount)) {
                status = IntegrationStatus.INTEGRATION_UNSUPPORTED;
            } else if (hasMissingIbanOrIntegrationFields(bankAccount, bankIntegration)) {
                status = IntegrationStatus.INTEGRATION_MISSING;
            } else {

```

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         | 101  |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         |      |

```

        status = testConnection(bankAccount, bankIntegration);
    }
    bankIntegration.setStatus(status);
    getBankIntegrationRepository().save(bankIntegration);
}

private boolean isBankSystem(BankAccountBaseInfoDto bankAccount) {
    return !bankAccount.getBank().getCode().isEmpty();
}

private boolean hasMissingIbanOrIntegrationFields(BankAccountBaseInfoDto bankAccount,
BankIntegration bankIntegration) {
    return isIbanEmpty(bankAccount) || hasAnyEmptyIntegrationField(bankIntegration.getFields());
}

private boolean isIbanEmpty(BankAccountBaseInfoDto bankAccount) {
    return Objects.isNull(bankAccount.getIban()) || bankAccount.getIban().isBlank();
}

private boolean hasAnyEmptyIntegrationField(List<BankIntegrationField> fields) {
    if (Objects.isNull(fields)) {
        return true;
    }
    return fields.stream()
        .anyMatch(field -> Objects.isNull(field.getValue()) || field.getValue().isBlank());
}

public ResponseEntity<?> synchronize(SynchronizeRequestDto synchronizeRequestDto) {
    getSynchronizationAsyncService().processSynchronization(synchronizeRequestDto);
    return ResponseEntity.ok().build();
}
}

@Service
@RequiredArgsConstructor
public class SynchronizationAsyncServiceImpl implements SynchronizationAsyncService {

    private final BankIntegrationRepository bankIntegrationRepository;
    private final RootClient rootClient;
    private final BankIntegrationHandler bankIntegrationHandler;
    private final AtomicBoolean isSynchronizing = new AtomicBoolean(false);

    @Async
    @Override
    public void processSynchronization(SynchronizeRequestDto dto) {
        if (!isSynchronizing.compareAndSet(false, true)) {
            return;
        }
        try {
            var bankIntegrations = bankIntegrationRepository.findAll();
            List<Long> bankAccountIds = dto.getBankAccountIds().isEmpty()
                ? bankIntegrations.stream()
                .map(BankIntegration::getBankAccountId)

```

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         | 102  |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         |      |

```

        .toList()
        : new ArrayList<>(dto.getBankAccountIds());
var bankAccountsMap = rootClient.fetchBankAccountsByIds(bankAccountIds).stream()
    .collect(Collectors.toMap(BankAccountBaseInfoDto::getId, b -> b));
bankIntegrations.forEach(bankIntegration -> {
    var bankAccount = bankAccountsMap.get(bankIntegration.getBankAccountId());
    var service =
bankIntegrationHandler.getServiceByBankCode(bankIntegration.getBankFields().getCode());
    if (bankAccount != null && bankAccount.getIban() != null) {
        var params = service.buildParams(bankIntegration, bankAccount);
        service.executeSynchronize(params, dto, bankAccount);
    }
});
} finally {
    isSynchronizing.set(false);
}
}

```

```

@Override
public SyncAvailabilityStatus getAvailabilityStatus() {
    if (isSynchronizing.get()) {
        return SyncAvailabilityStatus.ALREADY_IN_PROGRESS;
    }
    return bankIntegrationRepository.existsByStatus(IntegrationStatus.SUCCESSFULLY)
        ? SyncAvailabilityStatus.AVAILABLE
        : SyncAvailabilityStatus.NOT_AVAILABLE;
}
}

```

```

@Service("PRIVAT_BANK")
@RequiredArgsConstructor
@Slf4j
public class PrivateBankServiceImpl extends IntegrationBankService {

```

```

    private static final String ACC_KEY = "acc";
    private static final String CLIENT_ID_KEY = "clientId";
    private static final String TOKEN_KEY = "token";
    private static final String BANK_CODE = "PRIVAT_BANK";
    public static final Integer LIMIT = 100;
    public static final Integer TEST_LIMIT = 1;
    public static final String CONDUCTED = "r";
    public static final String CREDIT = "C";

```

```

@Getter
private final RootClient rootClient;
@Getter
private final BankIntegrationRepository bankIntegrationRepository;
@Getter
private final SynchronizationAsyncService synchronizationAsyncService;
private final PrivateBankClient privateBankClient;
private final ObjectMapper objectMapper;
private final TransactionsRepository transactionsRepository;

```

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         | 103  |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         |      |

```

private final TransactionsMapper transactionsMapper;

@Override
public String getBankCode() {
    return BANK_CODE;
}

@Override
public ResponseEntity<?> executeConnectionTest(Map<String, String> params) {
    try {
        var acc = IntegrationBankUtils.extractParams(params, ACC_KEY);
        var clientId = IntegrationBankUtils.extractParams(params, CLIENT_ID_KEY);
        var token = IntegrationBankUtils.extractParams(params, TOKEN_KEY);
        var queryParams = getBaseParam(acc);
        var response = privateBankClient.getTransactions(queryParams, token, clientId);
        return ResponseEntity.status(response.getStatusCode()).build();
    } catch (FeignException feignException) {
        return ResponseEntity.status(feignException.status()).build();
    }
}

@Override
@Transactional
public void executeSynchronize(Map<String, String> params, SynchronizeRequestDto
synchronizeRequestDto, BankAccountBaseInfoDto bankAccount) {
    var queryParams = getParams(synchronizeRequestDto,
IntegrationBankUtils.extractParams(params, ACC_KEY));
    fetchAllTransactions(queryParams, bankAccount,
        IntegrationBankUtils.extractParams(params, TOKEN_KEY),
        IntegrationBankUtils.extractParams(params, CLIENT_ID_KEY));
}

@Transactional
protected void fetchAllTransactions(GetTransactionsQueryParams queryParams,
BankAccountBaseInfoDto bankAccount, String token, String clientId) {
    String followId = null;
    var bankIntegration =
bankIntegrationRepository.getBankFieldsValueByBankAccountId(bankAccount.getId());
    do {
        setFollowId(queryParams, followId);
        try {
            updateBankIntegrationStatus(bankAccount, bankIntegration);
            var response = privateBankClient.getTransactions(queryParams, token, clientId);
            var transactionDto = objectMapper.convertValue(response.getBody(),
ResponseTransactionsDto.class);
            updateLastSynchronizedDate(bankIntegration);
            processTransactions(transactionDto, bankAccount);
            followId = getNextPageId(transactionDto);
        } catch (FeignException feignException) {
            handleFeignException(feignException, bankAccount, token, clientId, bankIntegration);
            return;
        }
    }
}

```

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         | 104  |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         |      |

```

    } while (Objects.nonNull(followId));
}

private void updateLastSynchronizedDate(BankIntegration bankIntegration) {
    bankIntegration.setLastSynchronizedAt(LocalDateTime.now());
    bankIntegrationRepository.save(bankIntegration);
}

private void setFollowId(GetTransactionsQueryParams queryParams, String followId) {
    if (Objects.nonNull(followId)) {
        queryParams.setFollowId(followId);
    }
}

private void updateBankIntegrationStatus(BankAccountBaseInfoDto bankAccount, BankIntegration
bankIntegration) {
    var integrationUnsupported = IntegrationBankUtils.getIntegrationStatus(bankAccount,
bankIntegration);
    if (Objects.nonNull(integrationUnsupported)) {
        bankIntegration.setStatus(integrationUnsupported);
    }
    bankIntegrationRepository.save(bankIntegration);
}

private String getNextPageId(ResponseTransactionsDto transactionDto) {
    return transactionDto.isExistNextPage() ? transactionDto.getNextPageId() : null;
}

private void handleFeignException(FeignException feignException, BankAccountBaseInfoDto
bankAccount, String token, String clientId, BankIntegration bankIntegration) {
    log.warn("Skipping transactions for bank account: {}, Token: {}, ClientId: {}. Response status:
{}",
        bankAccount.getId(), token, clientId, feignException.status());

    var statusCode = HttpStatusCode.valueOf(feignException.status());
    var integrationUnsupported = IntegrationBankUtils.getIntegrationStatusForCode(statusCode);
    bankIntegration.setStatus(integrationUnsupported);
    bankIntegrationRepository.save(bankIntegration);
}

@Transactional
protected void processTransactions(ResponseTransactionsDto transactionDto,
BankAccountBaseInfoDto bankAccount) {
    Optional.ofNullable(transactionDto.getTransactions())
        .ifPresent(transactions -> Arrays.stream(transactions)
            .filter(transaction -> !isTransactionProcessed(transaction.getTransactionId()))
            .filter(this::isConductedCreditTransaction)
            .forEach(transaction -> processTransaction(transaction, bankAccount)));
}

private boolean isConductedCreditTransaction(TransactionDetailsDto transaction) {
    return Objects.equals(transaction.getStatus(), CONDUCTED) &&

```

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         | 105  |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         |      |

```

        Objects.equals(transaction.getTransactionType(), CREDIT);
    }

    protected void processTransaction(TransactionDetailsDto transaction, BankAccountBaseInfoDto
bankAccount) {
        var studentId = IntegrationBankUtils.fetchStudentId(transaction.getPaymentPurpose());
        var student = studentId.map(rootClient::fetchStudent).orElse(null);

        try {
            rootClient.createOperation(buildOperationArriving(transaction, student, bankAccount));
            buildAndSaveTransaction(bankAccount, transaction, TransactionStatus.PROCESSED);
        } catch (Exception e) {
            log.error("Failed to create operation for transaction ID {}: {}", transaction.getTransactionId(),
e.getMessage());
            buildAndSaveTransaction(bankAccount, transaction, TransactionStatus.UNPROCESSED);
        }
    }

    private boolean isTransactionProcessed(String transactionId) {
        return transactionsRepository.existsByTransactionId(transactionId);
    }

    private WalletOperationDto buildOperationArriving(TransactionDetailsDto transactionDetailsDto,
StudentDto student,
BankAccountBaseInfoDto bankAccount) {

        var date =
IntegrationBankUtils.parseTransactionDate(transactionDetailsDto.getTransactionDate());
        var time =
IntegrationBankUtils.parseTransactionTime(transactionDetailsDto.getTransactionTime());
        var user = Objects.nonNull(student) ? buildUserInfo(student.getId()) : null;
        return WalletOperationDto.builder()
            .amount(transactionDetailsDto.getAmount())
            .bankAccount(buildBankAccountInfo(bankAccount))
            .user(user)
            .operationDate(date)
            .operationTime(time)
            .description(transactionDetailsDto.getPaymentPurpose())
            .paymentMethod(PaymentMethod.NON_CASH)
            .paymentType(PaymentType.NEW_PAYMENT)
            .build();
    }

    private BaseInfoDto buildBankAccountInfo(BankAccountBaseInfoDto bankAccount) {
        return BaseInfoDto.builder()
            .id(bankAccount.getId())
            .build();
    }

    private UserBaseInfoDto buildUserInfo(Long userId) {
        return UserBaseInfoDto.builder()
            .id(userId)
            .build();
    }

```

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         | 106  |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         |      |

```

    }

    private void buildAndSaveTransaction(BankAccountBaseInfoDto bankAccount,
                                         TransactionDetailsDto transactionDetailsDto,
                                         TransactionStatus status) {
        var transaction = transactionsMapper.toEntity(transactionDetailsDto);
        transaction.setBankAccountId(bankAccount.getId());
        transaction.setTransactionId(transaction.getTransactionId());
        transaction.setStatus(status);
        transactionsRepository.save(transaction);
    }

    @Override
    public Map<String, String> buildParams(BankIntegration bankIntegration,
                                           BankAccountBaseInfoDto bankAccount) {
        var params = bankIntegration.getFields().stream()
            .collect(Collectors.toMap(
                BankIntegrationField::getName,
                BankIntegrationField::getValue
            ));
        params.put(ACC_KEY, bankAccount.getIban());
        return params;
    }

    private GetTransactionsQueryParams getBaseParam(String acc) {
        var formatter = DateTimeFormatter.ofPattern("dd-MM-yyyy");
        var formattedDate = LocalDate.now().format(formatter);

        return GetTransactionsQueryParams.builder()
            .startDate(formattedDate)
            .endDate(formattedDate)
            .acc(acc)
            .limit(TEST_LIMIT)
            .build();
    }

    private GetTransactionsQueryParams getParams(SynchronizeRequestDto synchronizeRequestDto,
                                                  String acc) {
        var formatter = DateTimeFormatter.ofPattern("dd-MM-yyyy");
        var startDate = synchronizeRequestDto.getStartDate().format(formatter);
        var endDate = synchronizeRequestDto.getEndDate().format(formatter);
        return GetTransactionsQueryParams.builder()
            .startDate(startDate)
            .endDate(endDate)
            .acc(acc)
            .limit(LIMIT)
            .build();
    }
}

```

|      |      |                  |        |      |                                         |      |
|------|------|------------------|--------|------|-----------------------------------------|------|
|      |      | Садолінський В.А |        |      | «Житомирська політехніка».25.121.11.000 | Арк. |
|      |      | Громський О.О.   |        |      |                                         | 107  |
| Змн. | Арк. | № докум.         | Підпис | Дата |                                         |      |