

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:

«Вебплатформа для автоматизації роботи танцювальної студії»

Виконала студентка групи ІПЗ-22-3
СТУПАКОВА Анна Сергіївна

Керівник роботи:
ЧИЖМОТРЯ Олена Геннадіївна

Рецензент:
ЛОБАНЧИКОВА Надія Миколаївна

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

«__» лютого 2026 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **СТУПАКОВА Анна Сергіївна**

Керівник роботи: **ЧИЖМОТРЕ Олена Геннадіївна**

Тема роботи: **«Вебплатформа для автоматизації роботи танцювальної студії»**,

затверджена Наказом закладу вищої освіти від «27» лютого 2026 р., № 92/с

Вихідні дані для роботи: **діяльність танцювальної студії як об'єкт дослідження та вебплатформа для автоматизації управління**

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Чижмотря О.Г.	01.03.2026р.	01.03.2026р
2	Чижмотря О.Г.	01.03.2026р	01.03.2026р
3	Чижмотря О.Г.	01.03.2026р	01.03.2026р

РЕФЕРАТ

Випускна кваліфікаційна робота складається з вступу, трьох розділів, висновків, списку використаних джерел та додатків. Текстова частина викладена на 74 сторінках друкованого тексту. В роботі наведено 35 рисунків та 10 таблиць. Список використаних джерел містить 30 найменувань. Додатки займають 12 сторінки. Загальний обсяг роботи - 86 сторінок.

Метою роботи є розробка вебплатформи для автоматизації роботи танцювальної студії, яка забезпечує управління розкладом занять, облік учнів та викладачів, контроль відвідуваності та обробку фінансових операцій.

У роботі використано методи системного аналізу для дослідження предметної області, методи об'єктно-орієнтованого проєктування для побудови структури системи, методи реляційного моделювання для проєктування бази даних, а також методи вебпрограмування для реалізації платформи.

Результатом виконання роботи є розроблена вебплатформа, що автоматизує основні процеси діяльності танцювальної студії. Система реалізована на основі архітектурного шаблону MVC з використанням PHP та MySQL і забезпечує ведення розкладу занять, запис учнів, контроль відвідуваності, онлайн-оплату абонементів через платіжний сервіс LiqPay, програму лояльності, нарахування виплат викладачам та адміністрування всіх процесів студії.

Розроблена платформа може бути впроваджена в реальну діяльність танцювальних студій як готовий програмний продукт для автоматизації управлінських та організаційних процесів.

Ключові слова: ВЕБПЛАТФОРМА, ТАНЦЮВАЛЬНА СТУДІЯ, БАЗА ДАНИХ, PHP, MVC.

					ІПЗ.КР.Б – 121 – 26 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Вебплатформа для автоматизації роботи танцювальної студії	Літ.	Арк.	Аркуші
Розроб.		Ступакова А.С.						
Керівник		Чижмоторя О.Г.					3	86
Рецензент.		Лобанчикова Н.М.				Житомирська політехніка, група ІПЗ-22-3		
Зав. каф.		Вакалюк Т.А.						

ABSTRACT

The graduation qualification work consists of an introduction, three chapters, conclusions, a list of references, and appendices. The main text is presented on 74 pages of printed material. The work includes 35 figures and 10 tables. The list of references contains 30 sources. The appendices occupy 12 pages. The total volume of the work is 86 pages.

The aim of the work is to develop a web platform for automating the operations of a dance studio, which provides management of class schedules, student and teacher records, attendance tracking, and financial transaction processing.

The study employs system analysis methods to investigate the subject area, object-oriented design methods to build the system structure, relational modeling methods for database design, as well as web programming methods for implementing the platform.

The result of the work is a developed web platform that automates the main operational processes of a dance studio. The system is implemented based on the MVC architectural pattern using PHP and MySQL and provides class scheduling, student registration, attendance tracking, online subscription payments via the LiqPay payment service, a loyalty program, teacher payroll calculation, and administration of all studio processes.

The developed platform can be implemented in real dance studio operations as a ready-made software product for automating managerial and organizational processes.

Keywords: WEB PLATFORM, DANCE STUDIO, DATABASE, PHP, MVC.

		Ступакова А.С.			ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Чижмоторя О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		4

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ	6
ВСТУП	7
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ .	9
1.1. Постановка задачі	9
1.2. Аналіз аналогів програмного продукту	11
1.3. Вибір архітектури системи	15
1.4. Інструментальні засоби та вимоги до апаратних засобів	19
Висновки до першого розділу	23
РОЗДІЛ 2. РОЗРОБКА СТРУКТУРИ ТА ЛОГІКИ ФУНКЦІОНУВАННЯ СИСТЕМИ	25
2.1. Варіанти використання	25
2.2. Структура системи збереження даних та алгоритми роботи	29
2.3. Нормалізація та забезпечення цілісності даних	36
2.4. Структура модулів та класів системи	40
2.5. Опис реалізації ключових модулів системи	44
Висновки до другого розділу	49
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБПЛАТФОРМОЮ ТАНЦЮВАЛЬНОЇ СТУДІЇ	51
3.1. Порядок встановлення та налаштування системи	51
3.2. Інтерфейс користувача	53
3.3. Тестування системи	67
Висновки до третього розділу	70
ВИСНОВКИ	72
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	73
ДОДАТКИ	75

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

БД - це впорядковане сховище інформації, у якому зберігаються відомості про користувачів, заняття, розклад, абонементи та інші дані, необхідні для роботи програмної системи.

ПЗ - це сукупність програм, які забезпечують роботу вебплатформи, обробку даних та взаємодію користувачів із системою.

UML (Unified Modeling Language) - уніфікована мова моделювання, що використовується для графічного опису структури та поведінки програмної системи у вигляді діаграм, що полегшує проєктування та розуміння програмного забезпечення.

ERD (Entity-Relationship Diagram) - модель «сутність-зв'язок», що застосовується для проєктування бази даних і відображає основні сутності предметної області та зв'язки між ними.

PHP (Hypertext Preprocessor) - серверна мова програмування, що використовується для розробки серверної частини вебплатформи, обробки запитів користувачів, взаємодії з базою даних та реалізації основної логіки роботи системи.

MVC (Model-View-Controller) - це підхід до побудови програмних систем, який розділяє логіку обробки даних, інтерфейс користувача та керування взаємодією між ними, що спрощує розробку і супровід програмного забезпечення.

		Ступакова А.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

ВСТУП

Актуальність теми полягає в тому, що сьогодні багато сфер діяльності переходять у цифровий формат, проте в Україні танцювальні студії часто не мають зручних вебсайтів для організації своєї роботи. Зазвичай запис на заняття, розклад, оплати та спілкування з учнями відбуваються через соціальні мережі або месенджери, що не завжди є зручним і ефективним.

Під час пошуку подібних рішень було помічено, що в Україні майже відсутні сайти або вебплатформи, які спеціально створені для автоматизації роботи танцювальної студії. Саме тому виникла ідея розробити власну вебплатформу, яка дозволить об'єднати всі основні функції в одному місці та спростити роботу як для учнів, так і для викладачів та адміністрації.

Метою випускної кваліфікаційної роботи бакалавра є розробка та проєктування вебплатформи для автоматизації роботи танцювальної студії, яка дозволить спростити управління розкладом занять, записом учнів, відвідуваністю та оплатами.

Для реалізації поставленої мети потрібно виконати такі завдання:

- проведення аналізу предметної області діяльності танцювальної студії та визначення процесів, що підлягають автоматизації;
- аналіз існуючих аналогів програмного продукту та обґрунтування доцільності власної розробки;
- визначення архітектури системи та обґрунтування вибору інструментальних засобів реалізації;
- формування функціональних вимог та розробка UML-моделей системи;
- проєктування структури бази даних та нормалізація таблиць;
- реалізація програмного коду та розгортання готового програмного продукту;
- тестування реалізованого програмного комплексу.

		Ступакова А.С.			ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		7

Об'єктом дослідження є діяльність танцювальної студії, яка включає організацію навчального процесу, управління розкладом занять, облік відвідуваності учнів, взаємодію між учнями, викладачами та адміністрацією.

Предметом дослідження є вебплатформа для автоматизації роботи танцювальної студії, її структура, функціональні можливості та способи використання для підтримки основних організаційних процесів студії.

Методи дослідження. У роботі використано методи системного аналізу для дослідження предметної області та визначення вимог до системи, методи об'єктно-орієнтованого проєктування для побудови структури класів і модулів, методи реляційного моделювання для проєктування бази даних, а також методи вебпрограмування для реалізації серверної та клієнтської частин платформи.

Практичне значення отриманих результатів полягає в тому, що розроблена вебплатформа може бути впроваджена в реальну діяльність танцювальних студій для автоматизації управління розкладом, обліку учнів та викладачів, контролю відвідуваності та обробки платежів через інтегрований платіжний сервіс.

Структура та обсяг проєкту. Випускна робота складається з вступу, 3-х розділів, висновків, списку використаних джерел із 30 найменувань (2 стор.), 4 додатків (12 стор.), 10 таблиць та 35 ілюстрацій по тексту, з них 3 оформлена на окремій сторінці. Повний обсяг роботи становить 86 сторінок, з них 74 сторінок основного тексту.

		Ступакова А.С.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		8

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Постановка задачі

У процесі діяльності танцювальної студії виконується значна кількість організаційних, облікових та інформаційних процесів, пов'язаних із плануванням занять, записом учнів, обліком відвідуваності та здійсненням оплат. Крім того, до таких процесів також належать комунікація з клієнтами, інформування про зміни в розкладі, формування груп відповідно до рівня підготовки учнів, а також контроль завантаженості викладачів. У більшості випадків ці процеси реалізуються вручну або за допомогою розрізнених засобів, таких як соціальні мережі, месенджери чи електронні таблиці.

Такий підхід ускладнює управління роботою студії, призводить до втрати або дублювання інформації та знижує ефективність взаємодії між учнями, викладачами й адміністрацією. Зокрема, відсутність централізованого обліку оплат створює ризики фінансових помилок і ускладнює формування звітності, а ведення записів у месенджерах нерідко спричиняє накладки в розкладі, неповне відображення актуальної інформації та несвоєчасне інформування учнів про зміни. Також це призводить до значних витрат часу адміністратора на ручну обробку запитів і постійне оновлення даних у різних каналах одночасно.

Аналіз предметної області дозволив визначити основні задачі, які доцільно автоматизувати шляхом розробки спеціалізованої вебплатформи. Вибір саме вебзастосунку як форми реалізації обумовлений його кросплатформеністю - система буде однаково доступна з комп'ютера, планшета чи смартфона без необхідності встановлення додаткового програмного забезпечення, що особливо важливо для користувачів із різним рівнем технічної підготовки. Окрім цього, вебрішення спрощує оновлення системи та забезпечує централізоване зберігання даних.

		Ступакова А.С.			ІПЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		9

Автоматизація зазначених процесів сприятиме створенню єдиного інформаційного середовища, підвищенню точності обліку даних, зменшенню адміністративного навантаження та покращенню якості організації навчального процесу в танцювальній студії. Додатково це дозволить підвищити рівень прозорості фінансових операцій, забезпечити швидкий доступ до актуальної інформації та мінімізувати людський фактор у повсякденній роботі.

Платформа передбачає три основні ролі користувачів із чітко розмежованими правами доступу: клієнт, викладач та адміністратор. Клієнт взаємодіє з системою переважно для організації власного навчального процесу - перегляду розкладу, запису на заняття, скасування записів та контролю власних оплат і абонементів. Викладач зосереджений на проведенні занять, відмітці відвідуваності, комунікації з учнями, а також має доступ до особистої статистики та фінансової інформації щодо нарахованої винагороди. Адміністратор відповідає за операційне управління платформою в цілому - від керування користувачами та розкладом до контролю фінансових показників студії та формування звітності.

Розмежування прав доступу між ролями забезпечує коректну роботу кожного учасника в межах своїх повноважень та унеможливорює несанкціонований доступ до чужих даних. Це також підвищує рівень безпеки системи та дозволяє адаптувати інтерфейс під конкретні потреби кожного типу користувача, що покращує зручність використання платформи.

Перелік основних задач, що підлягають автоматизації, включає: аутентифікацію та авторизацію користувачів із розмежуванням прав доступу відповідно до ролей (учень, викладач, адміністратор); формування, перегляд та редагування розкладу занять; можливість онлайн-запису учнів на заняття з автоматичною перевіркою доступності місць; ведення обліку відвідуваності занять; управління абонементом та контроль оплат за заняття; збереження, оновлення та обробку інформації про учнів і викладачів; формування та перегляд статистики відвідуваності та фінансових показників; надсилання

		Ступакова А.С.			ІПЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		10

інформаційних повідомлень викладачем та адміністрацією щодо підтвердження запису, скасування занять або змін у розкладі.

Реалізація зазначених задач у межах вебплатформи створює передумови для ефективного управління діяльністю танцювальної студії та слугує основою для подальшого проєктування архітектури програмної системи, структури бази даних і модульної організації програмного комплексу.

1.2. Аналіз аналогів програмного продукту

Аналіз існуючих програмних та вебрішень, представлених на українському ринку, дозволяє оцінити сучасні підходи до організації діяльності танцювальних студій та автоматизації відповідних процесів. Незважаючи на те, що більшість студій мають власні вебсайти або сторінки в соціальних мережах, функціональність таких ресурсів здебільшого обмежується інформаційною складовою і не забезпечує повноцінної автоматизації управлінських та облікових процесів. Для аналізу було обрано три рішення, які можуть бути розглянуті як аналоги: вебсайти танцювальних студій Dance New Addiction, D.side Dance Studio, а також сервіс онлайн-запису EasyWeek.

Сайт танцювальної студії Dance New Addiction виконує роль інформаційного, функціонального вебресурсу та частково особистого блогу, який дозволяє користувачам ознайомитися з діяльністю студії, доступними напрямками занять і взаємодіяти з навчальним процесом в онлайн-форматі (Рис. 1.1).[6] Переваги та недоліки наведені у таблиці 1.1.

		Ступакова А.С.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		11

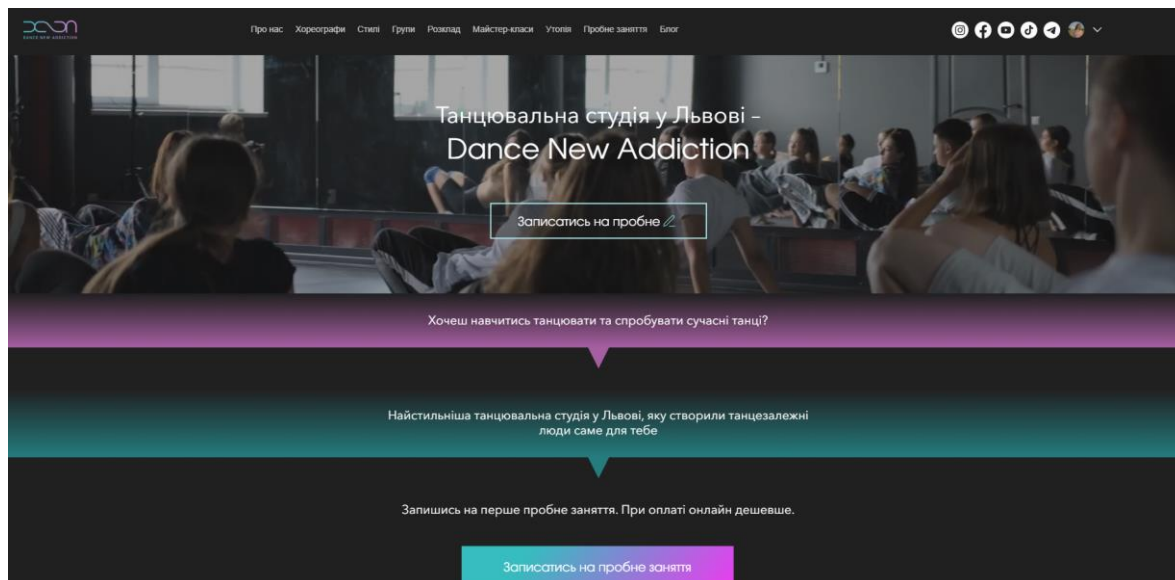


Рис. 1.1 - Головний екран студії Dance New Addiction

Таблиця 1.1

Переваги на недоліки Dance New Addiction

Переваги	Недоліки
наявна інформації про танцювальні напрямки, викладачів та розклад занять	дизайн сайту є зручним, проте не повністю відповідає сучасним вимогам вебдизайну
є можливість реєстрації користувачів та використання особистого кабінету	повільне завантаження сторінок та періодичні затримки в роботі сайту
також є особистий кабінет з можливістю запису на заняття	наявні функції не забезпечують повної автоматизації внутрішніх процесів студії
підтримка онлайн-оплати занять та абонементів через вбудований гаманець	можливість ведення власного блогу та підписки на інших користувачів, що заважає основному функціоналу сайту та ускладнює сайт

Вебсайт D.side Dance Studio орієнтований переважно на презентацію діяльності студії та інформування потенційних клієнтів про доступні танцювальні напрямки й формат занять. Ресурс виконує роль інформаційної платформи та не передбачає глибокої взаємодії користувачів із системою. Додаткові функції ведення блогу як для викладачів, так і для учнів, попри свою

		Ступакова А.С.			ІПЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		12

комунікативну цінність, розширюють платформу за межі її основного призначення. Такий підхід може збивати користувача з орієнтиру, оскільки ключова мета подібної системи - це запис на заняття та контроль навчального процесу. Другорядні функції перевантажують інтерфейс і розсіюють увагу від основних задач, що знижує зручність користування платформою. (Рис. 1.2).[7]

Переваги та недоліки наведені у таблиці 1.2.

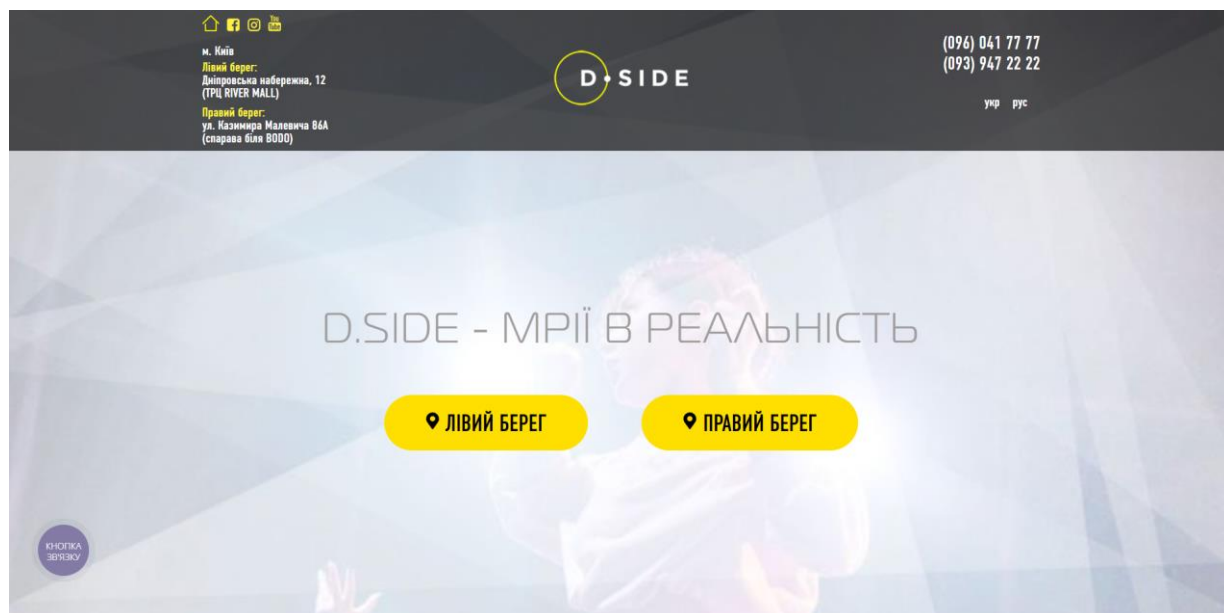


Рис. 1.2 - Головний екран студії D.side Dance Studio

Таблиця 1.2

Переваги на недоліки D.side Dance Studio

Переваги	Недоліки
детальний опис танцювальних напрямків і форматів тренувань	відсутній особистий кабінет як для клієнтів, так і для викладачів
наявність розкладу занять	можливість запису лише на пробне заняття
зручне та інтуїтивно зрозуміле ознайомлення з викладачами	відсутній автоматизований облік відвідуваності занять
привабливе візуальне оформлення	сайт не виконує функцій системи управління роботою танцювальної студії

Переваги	Недоліки
швидка робота сайту та оперативне завантаження сторінок	

EasyWeek є окремим онлайн-сервісом, який використовується для організації запису клієнтів та управління розкладом у різних сферах послуг. Даний сервіс не є сайтом конкретної студії, а виступає як універсальний інструмент для автоматизації окремих процесів (Рис. 1.3).[8] Переваги та недоліки наведені у таблиці 1.3.



Рис. 1.3 - Головний екран EasyWeek

Таблиця 1.3

Переваги на недоліки EasyWeek

Переваги	Недоліки
можливість онлайн-запису клієнтів на заняття	відсутність спеціалізації саме під діяльність танцювальних студій
роботу з розкладом занять у зручному календарному форматі	неможливість гнучкого налаштування функціоналу під конкретні потреби студії

Переваги	Недоліки
надсилання автоматичних повідомлень і нагадувань клієнтам	відсутність повноцінної системи роботи з абонементом та відвідуваністю
простий та зрозумілий інтерфейс	залежність від стороннього сервісу та його тарифної політики

Проведений аналіз показав, що жоден із розглянутих продуктів не є повноцінним рішенням для танцювальної студії. Сайти Dance New Addiction та D.side Dance Studio створені переважно для залучення нових клієнтів і не вирішують реальних управлінських задач. EasyWeek, хоч і автоматизує запис та розклад, є універсальним сервісом без прив'язки до специфіки танцювальної студії - у ньому відсутній облік відвідуваності, повноцінна робота з абонементом та облік виплат викладачам.

Розроблювана платформа відрізняється тим, що охоплює всі ці процеси в межах єдиної спеціалізованої системи без зайвого функціоналу. Студент може придбати абонемент через онлайн-оплату, записатися на заняття та відстежувати свою відвідуваність і накопичені бали лояльності. Викладач бачить свій розклад, відмічає присутніх та відстежує баланс нарахованої зарплатні. Адміністратор керує всіма процесами студії з єдиної панелі - від створення занять до виплат викладачам і модерації відгуків. Саме така повнота функціоналу в поєднанні з чіткою спеціалізацією є головною відмінністю розробки від існуючих аналогів.

1.3. Вибір архітектури системи

Проектування архітектури системи є одним із найважливіших етапів розробки програмного забезпечення. Саме на цьому етапі визначається, як будуть організовані всі частини програми, яким чином вони взаємодітимуть між собою та наскільки легко буде розширювати або змінювати систему в майбутньому. Правильно обрана архітектура - це фундамент, від якого

		Ступакова А.С.			ІПЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		15

залежать надійність роботи, зручність супроводу та здатність системи адаптуватися до нових вимог без повного переписування коду. Помилки, допущені на цьому етапі, важко виправити пізніше, оскільки вони впливають на всі рівні системи одночасно.

Система розроблена як веб-застосунок і побудована на основі моделі клієнт-сервер. Ця модель передбачає поділ на дві чіткі сторони: клієнтську і серверну. Клієнтська сторона - це те, що бачить і з чим безпосередньо взаємодіє користувач у браузері: сторінки сайту, форми, кнопки, таблиці та календарі. Серверна сторона залишається невидимою для користувача - вона отримує запити, перевіряє дані, виконує логіку програми, звертається до бази даних і повертає результат. Наприклад, коли студент натискає кнопку «Записатися на заняття», браузер надсилає запит на сервер. Сервер перевіряє, чи є вільні місця в групі, чи має студент активний абонемент, чи не записаний він уже на це заняття. Якщо всі умови виконані - запис фіксується в базі даних і браузер отримує підтвердження. Весь цей процес відбувається за частки секунди і є непомітним для користувача.

Перевага такого підходу полягає в тому, що система може одночасно обслуговувати багатьох користувачів з різними ролями - студентів, викладачів та адміністраторів. При цьому кожен отримує лише ті дані та можливості, які відповідають його ролі. Розподіл на клієнтську і серверну частини також спрощує подальший розвиток системи: зміни у зовнішньому вигляді сторінок не впливають на логіку обробки даних, і навпаки.

Для організації внутрішньої структури системи обрано архітектурний шаблон MVC, що розшифровується як Model-View-Controller — Модель, Подання, Контролер.[13] Вибір на користь цього шаблону зроблено після аналізу кількох альтернативних підходів та оцінки їх відповідності конкретним потребам проєкту. Найпростішою альтернативою є процедурний підхід, коли весь код пишеться послідовно у файлах без чіткого поділу на рівні - логіка роботи з даними, перевірки, формування відповіді і відображення перемішані в одному місці. Такий підхід зустрічається у старих або невеликих РНР-проєктах і може бути виправданий для простих сторінок із кількома

		Ступакова А.С.			ІПЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		16

функціями. Однак для системи управління студією, де є кілька ролей, десятки сторінок, складна бізнес-логіка та необхідність регулярного розширення, такий підхід швидко стає некерованим: зміна в одному місці ненавмисно ламає інше, а пошук і виправлення помилок перетворюються на трудомісткий процес.

Інша поширена альтернатива - мікросервісна архітектура, де система розбивається на кілька незалежних програм, кожна з яких відповідає за свою вузьку область і спілкується з рештою через мережу. Такий підхід добре зарекомендував себе у великих корпоративних системах із тисячами одночасних користувачів і командами розробників, що працюють паралельно над різними частинами продукту. Проте для системи рівня студії танців він є надлишковим: розгортання, підтримка і налагодження кількох взаємозалежних сервісів значно складніші, ніж єдиного застосунку, а виграш у масштабованості за такого обсягу даних і навантаження практично непомітний. Адміністративна складність такого рішення не виправдовує себе на даному етапі розвитку системи.

Також розглядався шаблон MVVM - Model-View-ViewModel, який широко використовується у сучасних JavaScript-фреймворках, таких як Vue або React, де інтерфейс будується як активний компонент із власним станом, що автоматично синхронізується з даними. Цей підхід ефективний для застосунків із багатою інтерактивністю на стороні браузера, де сторінка постійно оновлюється без перезавантаження. У даній системі переважна частина інтерфейсу формується на сервері і надсилається браузеру у готовому вигляді, тому впровадження MVVM вимагало б окремого клієнтського застосунку і значно ускладнило б архітектуру без відчутних переваг для кінцевого користувача.

MVC, натомість, є оптимальним вибором для серверних веб-застосунків із традиційною навігацією між сторінками. Цей шаблон є одним із найпоширеніших у веб-розробці та вирішує ключову проблему - запобігає змішуванню різних видів логіки в одному місці, що значно ускладнює підтримку та розвиток програми. Суть шаблону полягає в тому, що кожна

		Ступакова А.С.			ІПЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		17

частина системи виконує чітко визначену роль і не втручається в роботу інших частин.

Модель відповідає за дані та правила роботи з ними. Вона взаємодіє з базою даних - зберігає, шукає, змінює та видаляє інформацію - але не займається ні відображенням цієї інформації, ні обробкою дій користувача. У даній системі для кожної сутності існує окрема модель: модель користувачів, модель занять, модель абонементів, модель платежів, модель відвідуваності, модель відгуків та інші. Наприклад, модель занять вміє знайти всі заняття на поточний тиждень, перевірити наявність вільних місць у групі або повернути список студентів, записаних на конкретне заняття. Усі моделі мають спільний базовий клас, який забезпечує уніфікований і безпечний спосіб звернення до бази даних.

Подання - це те, що безпосередньо бачить користувач на екрані. Це сторінки сайту: розклад занять, особистий кабінет, форми для реєстрації та входу, адміністративна панель, картки викладачів, таблиці відвідуваності. Подання лише відображає готову інформацію, яку передає йому контролер, і не виконує жодних обчислень, перевірок або звернень до бази даних. Завдяки такому поділу зовнішній вигляд сторінки можна змінювати без будь-якого впливу на логіку роботи системи. При цьому кожна роль отримує власний варіант інтерфейсу з відповідним набором доступних функцій і відображуваних даних.

Контролер виконує роль посередника між користувачем і системою. Він отримує запит від браузера, визначає, яку дію потрібно виконати, звертається до відповідної моделі за даними або для їх збереження і передає результат поданням для відображення. Наприклад, коли адміністратор відкриває сторінку зі списком студентів, контролер отримує цей запит, просить модель дістати потрібні дані з бази і передає їх сторінці для показу. Окрім цього, контролер несе відповідальність за перевірку прав доступу: він перевіряє, чи авторизований користувач, яка в нього роль, і чи дозволено йому виконати запитану операцію. Якщо студент намагається відкрити адміністративну

		Ступакова А.С.			ІПЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		18

панель - контролер відмовляє у доступі ще до будь-якого звернення до бази даних.

Невід'ємною складовою архітектури є механізм централізованої обробки запитів. Усі запити від браузера, незалежно від адреси сторінки, надходять через єдину точку входу - один головний файл програми. Це забезпечує веб-сервер Apache за допомогою спеціальних правил переадресації. Такий підхід дозволяє виконувати єдиний набір перевірок для кожного запиту: перевірку авторизації, захист від підроблених форм, контроль розміру файлів, що завантажуються. Маршрутизатор - окремий компонент системи, що визначає, який контролер і який метод мають обробити конкретний запит, виходячи з адреси сторінки та типу дії.

Обрані архітектурні підходи у поєднанні формують систему з чітким поділом відповідальності між компонентами. Кожна частина системи є зрозумілою, ізольованою та замінною без необхідності переписувати решту програми. Таким чином, обрана архітектура формує надійну та логічно організовану основу для всієї системи управління танцювальною студією.

1.4. Інструментальні засоби та вимоги до апаратних засобів

Ефективна реалізація програмної системи неможлива без обґрунтованого вибору інструментальних засобів розробки, серед яких ключове місце займає мова програмування. Саме вона визначає підхід до побудови серверної логіки, рівень безпеки, продуктивність системи, а також зручність її підтримки та подальшого розвитку. Для веборієнтованих систем із клієнт-серверною архітектурою особливо важливо, щоб обрана мова забезпечувала стабільну обробку запитів, підтримку багатокористувацького доступу та можливість чіткої структуризації коду.

З урахуванням функціонального призначення системи, вимог до масштабованості та архітектурного підходу MVC було розглянуто кілька поширених мов програмування, які застосовуються для створення серверної

		Ступакова А.С.			ІПЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		19

частини вебдодатків. Подальший аналіз охоплює їхні переваги, недоліки та доцільність використання в межах даного проєкту.

PHP є однією з найбільш розповсюджених мов програмування для розробки серверної частини вебзастосунків.[14] Вона спочатку орієнтована на роботу з вебтехнологіями та забезпечує зручні механізми взаємодії з клієнтською частиною системи. Також характеризується високою продуктивністю та можливістю інтеграції з різноманітними вебсервісами і сторонніми бібліотеками. Значна кількість готових рішень та детальна документація дозволяють скоротити час розробки програмного забезпечення і спростити його подальшу підтримку. Порівняння наведено в таблиці 1.4.

Таблиця 1.4

Переваги та недоліки PHP

Переваги	Недоліки
нативна підтримка обробки HTTP-запитів і роботи з вебформами	за відсутності чіткої архітектури можливе зниження структурованості коду
простий та зрозумілий синтаксис	рівень безпеки залежить від коректності реалізації
повна сумісність із архітектурним шаблоном MVC	обмеження продуктивності у порівнянні з деякими компільованими мовами
широка підтримка на хостинг-платформах	
невисокі вимоги до апаратного забезпечення	

Python є популярною мовою програмування загального призначення, яка широко застосовується для розробки вебдодатків завдяки своїй простоті та високій читабельності коду.[18] Вона дозволяє швидко реалізовувати функціонал серверної частини та підтримує сучасні підходи до розробки

вебсистем, зокрема використання фреймворків, що прискорюють процес створення складних застосунків. Порівняння наведено в таблиці 1.5.

Таблиця 1.5

Переваги та недоліки Python

Переваги	Недоліки
висока читабельність і зрозуміла структура коду	складніше налаштування серверного середовища
швидка розробка функціоналу	вищі вимоги до апаратних ресурсів
вбудовані механізми безпеки та роботи з користувачами	нижча продуктивність у високонавантажених системах
наявність потужних вебфреймворків (Django, Flask)	менша поширеність на стандартних хостинг-платформах у порівнянні з PHP
зручність тестування та супроводу коду	

Java є об'єктно-орієнтованою мовою програмування, яка широко використовується для створення надійних та масштабованих серверних рішень.[27] Вона часто застосовується у великих корпоративних системах, де важливими є стабільність та довготривала підтримка. Порівняння наведено в таблиці 1.6.

Таблиця 1.6

Переваги та недоліки Java

Переваги	Недоліки
висока продуктивність і стабільність роботи	складніша реалізація та більший обсяг коду
добра масштабованість систем	триваліший процес розробки
сувора типізація, що зменшує кількість помилок	підвищені вимоги до апаратного забезпечення
розвинена екосистема фреймворків	надмірна складність для вебсистем середнього рівня

На основі проведеного аналізу було визначено, що PHP є найбільш доцільною мовою програмування для реалізації даної системи. Вона забезпечує оптимальне поєднання простоти розробки, гнучкості, продуктивності та відповідності архітектурному підходу MVC. Використання PHP дозволяє ефективно реалізувати необхідний функціонал, забезпечити стабільну роботу системи та спростити її подальший супровід і розвиток.

Для ефективної розробки та супроводу системи обрано Visual Studio Code як основне середовище розробки, що забезпечує повну інтеграцію з вебтехнологіями та підтримку сучасних стандартів програмування.[33] Серед ключових можливостей автозаповнення коду, підказки та рефакторинг, що прискорює розробку та знижує ймовірність помилок; інтеграція з git, що дозволяє ефективно працювати; вбудована підтримка HTML, CSS, JavaScript, що забезпечує комплексну розробку вебсистеми в одному середовищі; налагодження та тестування коду, що допомагає швидко виявляти і виправляти помилки.

Для збереження та обробки даних користувачів, розкладів занять, відвідуваності та фінансових операцій обрана реляційна база даних MySQL.[24] Основні переваги: висока продуктивність при роботі з великими обсягами даних; підтримка складних SQL-запитів і транзакцій, що забезпечує коректність операцій; сумісність із PHP і можливість інтеграції через стандартні бібліотеки; велика кількість документації та готових рішень.

Використання реляційної БД дозволяє організувати зберігання інформації у вигляді таблиць з чіткими зв'язками між ними, що спрощує доступ та обробку даних серверною частиною системи.

Отже для реалізації системи використані такі технології як мова програмування: PHP - для серверної логіки. Клієнтські технології HTML, CSS, JavaScript - для реалізації вебінтерфейсу та динамічної взаємодії з користувачем. Середовище розробки Visual Studio Code - для написання, налагодження та тестування коду.[30] Система контролю версій git - для управління змінами в коді. База даних MySQL - для зберігання та обробки даних користувачів і процесів студії.

		Ступакова А.С.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Чижмотря О.Г.				22
Змн.	Арк.	№ докум.	Підпис	Дата		

Цей набір інструментів забезпечує повний цикл розробки системи, від написання коду до тестування та підтримки роботи бази даних, а також дозволяє організувати якісний, надійний та масштабований продукт.

Вимоги до апаратного забезпечення

Система розрахована на роботу в багатокористувацькому середовищі, тому до апаратного забезпечення висуваються вимоги, що наведені у таблиці 1.7.

Таблиця 1.7

Серверна частина

Процесор	мінімум 4 ядра
Оперативна пам'ять	від 8 ГБ
Вільне місце на диску	від 100 ГБ
Підключення до мережі Інтернет	з пропускною здатністю не менше 100 Мбіт/с

Клієнтська частина має мати комп'ютер або ноутбук з сучасним браузером (Chrome, Firefox, Edge, Opera), мінімум 2 ГБ оперативної пам'яті та підключення до Інтернету.

Доступність та резервне копіювання бази даних та налаштувань системи має виконуватись регулярно автоматично раз на тиждень, та забезпечення безперервної роботи серверної частини з доступністю бази даних щонайменше 90%.

Такі вимоги гарантують стабільну роботу системи, підтримку одночасної роботи багатьох користувачів та можливість масштабування у майбутньому.

Висновки до першого розділу

У першому розділі проведено аналіз предметної області діяльності танцювальної студії, визначено задачі що підлягають автоматизації, обрано архітектуру системи та обґрунтовано вибір інструментальних засобів розробки.

		Ступакова А.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		23

Встановлено, що більшість організаційних процесів студії - планування занять, запис учнів, облік відвідуваності та контроль оплат - здійснюються вручну або за допомогою розрізаних засобів, що знижує ефективність управління та підвищує ризик втрати даних. Це обумовило необхідність розробки спеціалізованого програмного рішення.

Проведено аналіз існуючих аналогів: сайтів танцювальних студій Dance New Addiction та D.side Dance Studio, а також сервісу онлайн-запису EasyWeek. Аналіз показав, що жоден із розглянутих продуктів не забезпечує комплексної автоматизації всіх ключових процесів студії. Інформаційні сайти студій виконують переважно презентаційну функцію, а Dance New Addiction додатково обтяжений непотрібним соціальним функціоналом. Універсальні сервіси на кшталт EasyWeek не враховують специфіку навчального процесу та не підтримують облік відвідуваності, роботу з абонементом чи виплати викладачам. Це підтверджує доцільність розробки власної спеціалізованої вебплатформи.

Для реалізації системи обрано архітектурний шаблон MVC, який забезпечує чітке розмежування між даними, логікою та інтерфейсом користувача. Як мову серверної розробки обрано PHP, що найкраще відповідає вимогам проєкту за простотою, гнучкістю та сумісністю з обраною архітектурою. Для зберігання даних використовується MySQL, для написання коду — Visual Studio Code з підтримкою git.

Отримані результати стали основою для подальшого проєктування структури бази даних, розробки алгоритмів роботи системи та реалізації програмного комплексу.

		Ступакова А.С.			ІПЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		24

РОЗДІЛ 2. РОЗРОБКА СТРУКТУРИ ТА ЛОГІКИ ФУНКЦІОНУВАННЯ СИСТЕМИ

2.1. Варіанти використання

Для забезпечення ефективної роботи танцювальної студії, спрощення процесів управління та покращення взаємодії між учнями, викладачами й адміністрацією було проведено аналіз вимог до програмного продукту. Ці вимоги спрямовані на автоматизацію ключових процесів студії, підвищення зручності користування системою та забезпечення її стабільної роботи.

Результати аналізу вимог користувачів дали змогу визначити основні варіанти використання системи. На їх основі формується UML-діаграма варіантів використання платформи танцювальної студії в Додатку А.[21]

У системі розрізняють трьох основних акторів: адміністратор, викладач та клієнт.

Короткий опис акторів представлено в табл. 2.1.

Таблиця 2.1

Виявлення акторів

Актор	Короткий опис
Адміністратор	Керує всією системою: створює та редагує абонементи, заняття, викладачів, учнів, керує розкладом, системою лояльності, оплатами та має доступ до всієї інформації платформи.
Викладач	Проводить заняття, веде облік відвідуваності учнів, переглядає розклад, має доступ до інформації про свої групи та учнів.
Клієнт	Переглядає розклад занять, записується на заняття, оплачує абонементи, отримує нагадування, переглядає історію відвідувань, залишає відгуки та користується системою лояльності.

На основі визначених ролей було виконано виявлення варіантів використання системи, які описують функціональні можливості платформи та взаємодію кожного актора з нею. Дані варіанти використання згруповано відповідно до ролей користувачів і наведено в таблиці 2.2.

Таблиця 2.2

Виявлення варіантів використання

Основний актор	Найменування	Формулювання
Клієнт / Викладач	Авторизація	Дає змогу увійти в систему, використовуючи власні облікові дані.
	Реєстрація	Дозволяє створити новий обліковий запис у системі.
	Редагування особистих даних	Дає змогу змінювати персональну інформацію користувача.
Користувач	Запис на заняття	Дозволяє здійснювати онлайн-запис на групові або індивідуальні заняття.
	Перегляд розкладу	Надає доступ до актуального розкладу занять та афіші подій студії.
	Оплата абонементів	Дозволяє оплачувати абонементи через онлайн-платформу.
	Перегляд історії відвідувань	Дає змогу переглядати інформацію про відвідані заняття.

Основний актор	Найменування	Формулювання
	Отримання нагадувань	Дозволяє отримувати повідомлення про запис на заняття заняття або зміни в розкладі.
	Залишення відгуків	Надає можливість залишати відгуки про викладачів.
Викладач	Ведення обліку відвідуваностім	Дозволяє фіксувати присутність учнів на заняттях.
	Перегляд інформації про учнів	Дає доступ до списків учнів та розкладу занять.
	Надсилання важливих сповіщень групі	Дозволяє надсилати повідомлення учням групи
	Відслідковувати свій баланс	Дозволяє бачити скільки коштів на особистому рахунку і потім отримати виплату
Адміністратор	Керування розкладом	Дозволяє створювати, редагувати та видаляти заняття.
	Керування викладачами та клієнтами	Забезпечує створення, редагування або блокування облікових записів.
	Керування платежами та лояльністю	Дозволяє керувати платежами, акціями, бонусами та знижками.
	Виплачувати викладачам зарплату	Дозволяє робити виплати викладачам в залежності скільки клієнтів у нього було та який у нього відсоток

Основний актор	Найменування	Формулювання
	Отримання статистики	Надає доступ до статистики відвідуваності та оплат.

Функціональні вимоги визначають конкретні можливості, які система повинна забезпечувати для кожної ролі користувача.

Система повинна забезпечувати реєстрацію та авторизацію користувачів з розмежуванням прав доступу відповідно до ролі - студент, викладач або адміністратор. Для студентів передбачена можливість як повноцінної реєстрації з паролем, так і спрощеного гостьового запису лише за іменем і номером телефону для відвідування пробного заняття.

Система повинна дозволяти студентам переглядати розклад занять із фільтрацією за стилем, викладачем, залом та датою, записуватися на заняття за наявності активного абонементу, а також скасовувати запис. Оплата абонементів здійснюється онлайн через інтегрований платіжний сервіс LiqPay, після чого абонемент автоматично активується або ставиться в чергу, якщо у студента вже є активний.

Система повинна автоматично списувати одне заняття з абонементу після фіксації присутності студента, а у разі вичерпання поточного абонементу - автоматично активувати наступний з черги. За кожне відвідане заняття студенту нараховуються бали програми лояльності, які він може обміняти на безкоштовне заняття після досягнення встановленого порогу.

Викладач повинен мати можливість переглядати свій розклад, відмічати присутність студентів на заняттях та відстежувати власний баланс нарахованої винагороди. Система повинна автоматично розраховувати розмір заробітної плати викладача за кожне відвідане студентом заняття на основі індивідуального відсотка ставки.

Адміністратор повинен мати доступ до повного управління системою: створення та редагування занять із перевіркою конфліктів розкладу, управління обліковими записами студентів і викладачів, перегляд платежів та

		Ступакова А.С.			ІПЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		28

формування виплат викладачам, а також модерація відгуків і перегляд статистики відвідуваності та фінансів.

Нефункціональні вимоги визначають якісні характеристики системи, що забезпечують її надійність, безпеку та зручність використання.

З точки зору безпеки система повинна забезпечувати захист від міжсайтових підробок запитів шляхом перевірки токена при кожному POST-запиті. Паролі користувачів повинні зберігатися виключно у хешованому вигляді та ніколи не передаватися у відкритому вигляді. Усі дані, що надходять від користувача через форми, повинні проходити валідацію та санітизацію перед обробкою.

З точки зору доступності інтерфейс системи повинен коректно відображатися у сучасних браузерях та забезпечувати зручну роботу як з комп'ютера, так і з мобільного пристрою. Система повинна підтримувати одночасну роботу кількох користувачів без втрати продуктивності та узгодженості даних.

З точки зору надійності резервне копіювання бази даних повинно виконуватися автоматично не рідше одного разу на тиждень. Доступність серверної частини повинна становити не менше 90% часу. Усі критичні операції, такі як списання занять з абонементу та нарахування виплат, повинні виконуватися без можливості подвійного застосування однієї й тієї самої операції.

2.2. Структура системи збереження даних та алгоритми роботи

Ефективне функціонування програмного комплексу танцювальної студії потребує чітко спроектованої структури збереження даних та формалізованих алгоритмів реалізації основних бізнес-процесів. Логічна модель даних відображає сутності предметної області та їх взаємозв'язки, а алгоритми роботи системи визначають послідовність виконання операцій під час взаємодії користувачів із програмним комплексом.

		Ступакова А.С.			ІПЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		29

Система збереження даних побудована на основі реляційної моделі, що забезпечує цілісність, узгодженість та контроль залежностей між об'єктами. Для візуального представлення структури бази даних використано Entity-Relationship Diagram (ERD), на якій показані всі сутності, їхні атрибути та типи зв'язків між ними. Основних сутності системи наведені у таблиці 2.3.

Таблиця 2.3

Основні сутності системи

Сутність	Опис
users	обліковий запис користувача (клієнт, інструктор, адміністратор), який визначає роль і доступ до функціоналу системи
teacher_profile	розширений профіль викладача: біографія, аватар, відсоток ставки винагороди та посилання на категорію, профіль створюється адміністратором одночасно з обліковим записом викладача
teacher_grades	довідник категорій викладачів («Початківець», «Середній рівень», «Досвідчений», «Майстер») із відповідним відсотком ставки
dance_styles	перелік усіх стилів танців студії
halls	зали студії, що використовуються при плануванні занять
subscription_types	типи абонементів з ціною, кількістю занять, терміном дії та ознакою активності

Сутність	Опис
subscriptions	конкретна покупка абонементу клієнтом, зберігає поточний залишок занять, статус і дату завершення. Якщо клієнт купує новий абонемент при наявному активному, він отримує статус pending і автоматично активується після вичерпання попереднього
payments	запис про платіж із зовнішнім ідентифікатором транзакції, методом оплати, сумою, валютою та статусом, прив'язується до типу абонементу та до конкретного абонементу після успішної оплати.
classes	одиничне заняття із зазначенням стилю, викладача, залу, рівня складності, часу початку і завершення, максимальної кількості учасників і статусу
class_enrollments	запис студента на конкретне заняття, що містить статус запису, прив'язку до абонементу, ознаку розрахованості та ознаку пробного заняття
attendance	факт відвідування конкретного запису на заняття з різним статусом

Сутність	Опис
teacher_payout_lines	рядки нарахування винагороди викладачу за кожне відвідане студентом заняття
teacher_payment_batches	пакети виплат викладачам, що об'єднують окремі рядки нарахування та фіксують загальну суму й дату виплати
reviews	відгуки студентів про викладачів з рейтингом (1–5), коментарем та ознакою публікації
loyalty_points	бали програми лояльності, де за кожне відвідане заняття студент отримує фіксовану кількість балів, а при накопиченні порогового значення можна обміняти їх на безкоштовне заняття
blog_posts	статті блогу студії з заголовком, вмістом, обкладинкою, автором та статусом публікації

Між сутностями реалізовано різні типи зв'язків, що забезпечують узгодженість даних. Один користувач із роллю викладача має рівно один розширений профіль та може бути пов'язаний із кількома стилями танців через відповідну зведену таблицю. Один студент може мати кілька абонементів одночасно - активний та один або інші у черзі на активацію, а також кілька записів на різні заняття. Кожне заняття прив'язане до одного танцювального стилю, одного викладача та одного залу, при цьому перед збереженням заняття система автоматично перевіряє відсутність конфліктів за часом як для викладача, так і для залу. Запис на заняття є сполучною ланкою між студентом,

заняттям та абонементом, тому саме на його основі фіксується відвідуваність, нараховується винагорода викладачу та бали програми лояльності студенту. Відгук прив'язаний до конкретного запису на заняття і може бути залишений лише за умови фактичного відвідування, тобто за наявності підтвердженого запису про присутність. Рядки нарахування винагороди посилаються на конкретний запис, заняття та студента, що дозволяє адміністратору переглядати повну деталізацію нарахувань і формувати виплатні пакети для викладачів.

Для відображення поточних станів об'єктів у системі використовуються фіксовані переліки допустимих значень. Роль користувача визначає рівень доступу і може набувати значень: студент, викладач або адміністратор. Стан абонементу відображає його життєвий цикл: активний означає, що абонемент діє і заняття доступні; очікує активації - абонемент придбано, але він стоїть у черзі за поточним активним; вичерпано - всі заняття використано; прострочено - термін дії минув; призупинено - абонемент тимчасово заблоковано адміністратором. Стан платежу відображає результат обробки транзакції: очікує підтвердження від платіжного сервісу, успішно оплачено або відхилено. Стан запису на заняття показує, на якому етапі знаходиться участь студента: записаний, відвідав, скасовано студентом або скасовано адміністратором. Стан відвідуваності фіксується викладачем або адміністратором і може бути: присутній, відсутній або відсутній з поважної причини.

Повну Entity-Relationship Diagram (ERD), що відображає всі сутності, їхні поля та взаємозв'язки (Рис. 2.1).

		Ступакова А.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		33

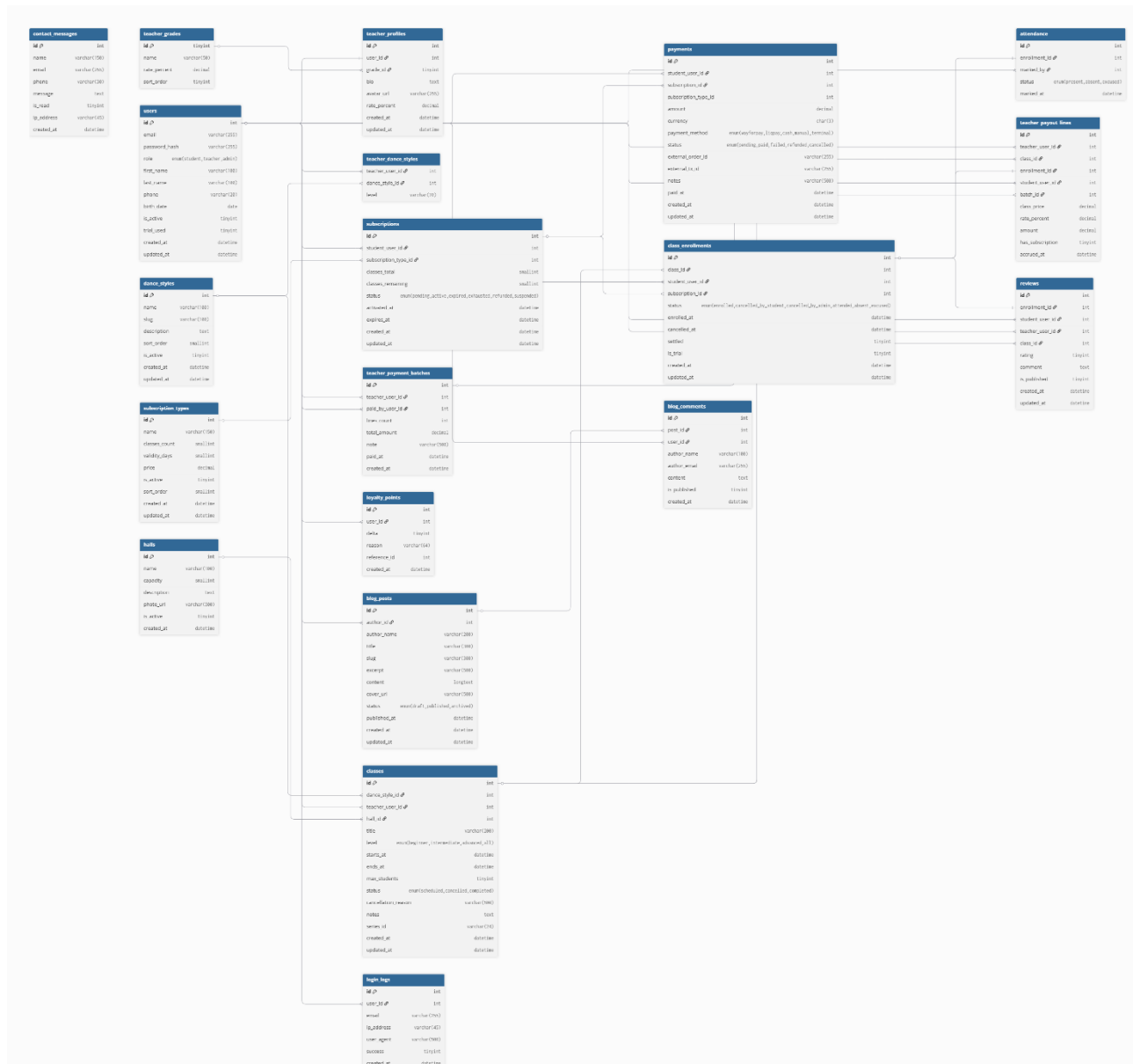


Рис. 2.1 - Entity-Relationship Diagram

Алгоритми роботи системи формалізують основні процеси функціонування танцювальної студії та забезпечують коректну взаємодію між користувачами і внутрішніми компонентами програмного комплексу.

Авторизація та реєстрація реалізовані стандартним чином: студент заповнює форму реєстрації або входить за електронною поштою і паролем, система перевіряє дані та надає доступ до відповідного кабінету. Облікові записи викладачів створюються виключно адміністратором.

Оформлення та оплата абонементу є одним із ключових алгоритмів системи. Студент обирає тип абонементу на відповідній сторінці, після чого система формує запис платежу зі статусом «очікує підтвердження» та генерує

унікальний зовнішній ідентифікатор замовлення. Далі користувача перенаправляють на платіжну форму LiqPay.[12, 29] Після завершення транзакції платіжний сервіс надсилає підписаний callback-запит на окремий ендпоінт системи. Система перевіряє цифровий підпис запиту, знаходить відповідний платіж за зовнішнім ідентифікатором і оновлює його статус. У разі успішної оплати статус платежу змінюється на «оплачено», після чого система автоматично створює запис абонементу для студента. При цьому виконується перевірка: якщо у студента вже є активний абонемент, новий отримує статус «очікує активації» і ставиться в чергу; якщо активного абонементу немає, тоді новий одразу активується із розрахунком дати завершення на основі терміну дії типу абонементу. У разі відхилення платежу або помилки його статус встановлюється як відхилено, абонемент не створюється, а студент отримує відповідне повідомлення.

Запис на заняття та облік відвідуваності є найскладнішим алгоритмом системи, оскільки він охоплює кілька взаємопов'язаних перевірок і тригерів. Коли студент намагається записатися на заняття, система послідовно виконує такі перевірки: наявність активного абонементу із залишком занять більше нуля, відсутність вже існуючого запису на це заняття, а також відсутність накладання за часом з іншими заняттями студента. Якщо всі умови виконано, тоді створюється запис зі статусом записаний із прив'язкою до абонементу. Для студентів без абонементу передбачена можливість одного пробного заняття: система перевіряє, що прапорець пробного заняття ще не використано, і створює запис із відповідною ознакою.

Після проведення заняття викладач або адміністратор відмічає кожного студента у списку. Коли фіксується присутність, система виконує ланцюжок операцій: вносить запис про відвідування, списує одне заняття з абонементу студента, зменшуючи лічильник залишку. Якщо після списання залишок досягає нуля, абонемент отримує статус «вичерпано», а система автоматично перевіряє наявність наступного абонементу в черзі та активує його, розраховуючи нову дату завершення. Паралельно система нараховує рядок виплати викладачу: розраховує ціну одного заняття шляхом ділення вартості

		Ступакова А.С.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		35

абонементу на кількість занять у ньому, множить її на індивідуальний відсоток ставки викладача та зберігає результат. Унікальний індекс на ідентифікаторі запису унеможливорює подвійне нарахування. Якщо студент був відсутній з поважної причини — заняття не списується, але запис позначається як розрахований. Для пробних занять логіка спрощена: заняття з абонементу не списується, але нарахування викладачу все одно відбувається.

Програма лояльності. Одночасно з нарахуванням виплати викладачу система нараховує студенту бал лояльності за відвідане заняття. Коли накопичений баланс балів досягає встановленого порогу, студент може обміняти їх на безкоштовне заняття у відповідному розділі кабінету. Система списує порогову кількість балів і записує студента на обране заняття без прив'язки до платного абонементу.[10]

Написання відгуків. Студент може залишити відгук про викладача лише після підтвердженого відвідування хоча б одного його заняття та за умови відсутності попереднього відгуку. Відгук зберігається зі статусом «не опубліковано» і стає видимим після модерації адміністратором.

Управління розкладом. Адміністратор створює заняття з вибором викладача, залу, стилю та часового проміжку. Перед збереженням система перевіряє відсутність конфліктів: чи не зайнятий викладач в цей час іншим заняттям і чи вільний обраний зал. Підтримується також створення серій занять, що повторюються, з можливістю групового скасування.

2.3. Нормалізація та забезпечення цілісності даних

Проектування бази даних вебплатформи танцювальної студії здійснюється з дотриманням принципів нормалізації реляційних баз даних. Нормалізація - це процес організації таблиць і зв'язків між ними таким чином, щоб усунути надмірність зберігання інформації, запобігти аномаліям при додаванні, оновленні та видаленні даних, а також забезпечити логічну узгодженість структури.[1, 11] Ненормалізована база даних може призвести до ситуацій, коли одна і та сама інформація зберігається в кількох місцях, і зміна

		Ступакова А.С.			ІПЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		36

в одному місці не відображається в іншому, що рано чи пізно призводить до суперечностей і некоректної роботи системи.

Перша нормальна форма вимагає, щоб кожна клітинка таблиці містила лише одне атомарне значення, а не список чи набір даних. Наприклад, інформація про стилі танців, якими володіє викладач, не зберігається у вигляді переліку в одному полі таблиці викладача, а винесена в окрему зведену таблицю, де кожен запис відповідає одному стилю одного викладача із зазначенням рівня його підготовки - початківець, середній або просунутий рівень. Якби всі стилі зберігалися одним рядком в одному полі, пошук усіх викладачів певного стилю або оновлення одного зі стилів були б значно складнішими і ненадійнішими операціями. Схожим чином організоване зберігання записів студентів на заняття, де кожен запис є окремим рядком у відповідній таблиці, що дозволяє точно відстежувати статус кожного запису незалежно від інших. Кожна таблиця має первинний ключ - унікальний ідентифікатор, що однозначно ідентифікує кожен запис і не може повторюватися.

Друга нормальна форма вимагає, щоб кожне поле таблиці залежало від усього первинного ключа, а не лише від його частини. Ця вимога стосується передусім таблиць зі складеним ключем, тобто таких, де первинний ключ утворений комбінацією кількох полів. Зокрема, зведена таблиця зв'язку викладачів зі стилями танців містить лише ті атрибути, які безпосередньо стосуються обох частин ключа - і конкретного викладача, і конкретного стилю одночасно. Якби в цій таблиці зберігалось ім'я викладача, це було б порушенням другої нормальної форми, адже ім'я залежить лише від викладача, а не від комбінації викладача зі стилем. Жодних часткових залежностей у структурі бази даних немає, що забезпечує коректність даних при будь-яких операціях.

Третя нормальна форма вимагає усунення транзитивних залежностей, коли одне неключове поле залежить не від ключа таблиці, а від іншого неключового поля. Найяскравіший приклад у проєкті є розділення даних користувача і специфічних даних викладача. Біографія, аватар та відсоток

		Ступакова А.С.			ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		37

ставки стосуються лише викладачів, тому зберігати їх разом із загальними даними всіх користувачів системи було б некоректно. Якщо завтра в системі з'явиться нова категорія користувачів, ці поля залишалися б порожніми для них, що засмічує структуру і ускладнює підтримку. Тому ці дані винесено в окрему таблицю профілів викладачів, кожен запис якої пов'язаний рівно з одним користувачем з відповідною роллю.

Схожим чином вирішено питання з абонементом. Параметри типу абонементу - ціна, кількість занять, термін дії, назва зберігаються в окремій таблиці типів, тоді як факт покупки конкретного абонементу конкретним студентом зберігається окремо і лише посилається на відповідний тип. Якщо десятки студентів придбають однаковий абонемент, інформація про його умови зберігається в одному місці, а не дублюється в кожному записі. Це означає, що якщо студія вирішить змінити ціну або кількість занять в абонементі, достатньо оновити один запис у таблиці типів - всі нові покупки одразу відображатимуть актуальні умови без жодних додаткових дій. Категорії викладачів також виділено в окрему таблицю: зміна відсоткової ставки для певної категорії автоматично відображається для всіх викладачів цієї категорії без необхідності оновлювати кожен запис окремо. Аналогічно таблиця відвідуваності є окремою і лише пов'язана із записами студентів на заняття, що дозволяє зберігати повну інформацію про кожне відвідування - хто позначив, коли і з якою приміткою, не захаращуючи основну таблицю записів.

Окрім нормалізації структури, у системі реалізовано кілька механізмів, що забезпечують цілісність даних під час роботи. Перший і найважливіший з них - зовнішні ключі на рівні бази даних. Зовнішній ключ - це обмеження, яке гарантує, що запис може посилатися лише на реально існуючий об'єкт в іншій таблиці.[22] Не можна, наприклад, записати студента на заняття, якого немає в системі, або нарахувати виплату за відвідування, якого не існує. Якщо заняття видаляється з системи, всі записи студентів на це заняття видаляються автоматично - база даних сама підтримує узгодженість, не покладаючись на те, що програмний код не забуде виконати це вручну. В інших випадках, наприклад при видаленні підписки, поле посилання на неї в записі про оплату

		Ступакова А.С.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		38

не видаляється, а лише обнуляється - це дозволяє зберегти історію платежів навіть після закінчення дії абонементу.

Унікальні індекси захищають від дублювання критичних даних. Електронна адреса та номер телефону студента є унікальними - один і той самий контакт не можна зареєструвати двічі. Це запобігає ситуаціям, коли один користувач випадково або навмисно реєструється повторно і отримує незалежний профіль. Унікальний індекс у таблиці записів студентів на заняття гарантує, що один студент не може записатися на одне і те ж заняття двічі, навіть якщо два запити надійдуть одночасно. Аналогічно унікальний індекс у таблиці нарахувань виплат гарантує, що за одне відвідування викладач отримає нарахування рівно один раз, навіть якщо відповідна операція з технічних причин буде викликана повторно.

На рівні програмної логіки система перевіряє умови перед виконанням будь-якої важливої операції. Перед записом студента на заняття перевіряється наявність активного абонементу, кількість залишених занять, відсутність конфлікту за часом з іншими записами, а також наявність вільних місць у групі. Перед позначенням відвідуваності перевіряється, чи настав час заняття. Перш ніж студент зможе залишити відгук про викладача, система перевіряє, чи дійсно цей студент відвідував заняття саме цього викладача - без цього залишити відгук неможливо. Такий підхід не дозволяє даним потрапити в суперечливий або логічно некоректний стан навіть у разі некоректних дій з боку користувача.

Усі дані, що надходять від користувача через форми, проходять перевірку формату та очищення від потенційно небезпечних символів. Кожен POST-запит захищений унікальним токеном, який генерується для кожного сеансу користувача і перевіряється перед виконанням будь-якої дії - це унеможливорює підроблення запитів від імені користувача зі сторонніх сайтів. Паролі користувачів зберігаються виключно у хешованому вигляді за алгоритмом bcrypt, що унеможливорює їх відновлення навіть у разі витоку бази даних.[25] У сукупності всі ці механізми утворюють багаторівневий захист, де кожен рівень доповнює інший і компенсує можливі недоліки суміжних рівнів.

		Ступакова А.С.			ІПЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Чижомотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		39

2.4. Структура модулів та класів системи

Структура інформаційної системи танцювальної студії побудована за принципами об'єктно-орієнтованого програмування та реалізована відповідно до архітектурного шаблону MVC. Суть цього підходу полягає в тому, що система чітко поділяється на три незалежні рівні: моделі відповідають за роботу з даними та бізнес-логіку, контролери обробляють запити користувача і керують взаємодією між компонентами, а шаблони формують те, що в результаті бачить користувач у браузері. Завдяки такому розподілу кожен частину системи можна змінювати незалежно, не торкаючись решти коду. Наприклад, повне оновлення зовнішнього вигляду адміністративної панелі не вимагає жодних змін у логіці роботи з базою даних, а вдосконалення алгоритму нарахування балів лояльності не впливає на те, як виглядає кабінет студента. Для наочного представлення структури в роботі використано діаграму класів предметної області (Додаток Б) та архітектурну діаграму в межах MVC (Додаток В).

Перш ніж розглядати окремі рівні, варто описати базові класи ядра, на яких тримається вся система. Перший з них відповідає за маршрутизацію: коли користувач відкриває певну сторінку або надсилає форму, система зіставляє адресу запиту зі списком зареєстрованих маршрутів і визначає, який контролер і який метод має обробити цей запит. Маршрути підтримують динамічні параметри - наприклад, адреса сторінки конкретного викладача містить його ідентифікатор, який автоматично передається у відповідний метод. Окрім цього, маршрутизатор перевіряє захисний токен для кожної форми, що унеможливорює підроблення запитів від імені користувача зі сторонніх сайтів. Єдиний виняток - зовнішній запит від платіжної системи, який надходить не від браузера користувача, а напряму від платіжного сервісу і тому перевіряється за окремим механізмом підпису.

Другий базовий клас ядра забезпечує підключення до бази даних і реалізований за шаблоном проєктування «Одинак» (Singleton). Суть цього шаблону полягає в тому, що клас гарантує існування лише одного власного

		Ступакова А.С.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		40

екземпляра впродовж усього циклу виконання програми і надає єдину глобальну точку доступу до нього.[28] Практично це означає наступне: коли перша модель під час обробки запиту звертається до бази даних, клас перевіряє, чи існує вже встановлене з'єднання. Якщо ні - створює його і зберігає. Коли наступна модель робить своє звернення, клас повертає вже існуюче з'єднання без жодних додаткових витрат. Таким чином, скільки б моделей не звернулося до бази даних в рамках одного запиту, фізичне підключення до сервера встановлюється рівно один раз. Це суттєво зменшує навантаження на сервер бази даних, адже встановлення нового з'єднання є відносно дорогою операцією з точки зору часу та ресурсів. Додатковою перевагою є те, що всі операції в рамках одного запиту використовують одне й те саме з'єднання, що дозволяє виконувати кілька пов'язаних операцій як єдине ціле - якщо одна з них завершиться невдало, решта може бути скасована автоматично. Налаштування підключення читаються зі змінних оточення, що дозволяє використовувати різні бази даних для розробки, тестування та промислового середовища без зміни коду.

Батьківські класи для контролерів і моделей визначають спільний набір можливостей, які автоматично доступні всім дочірнім класам. Базовий контролер надає методи для відображення шаблонів із автоматичним додаванням захисного токена, для перенаправлення на інші сторінки, для перевірки прав доступу залежно від ролі користувача, а також для безпечного отримання даних із форм із очищенням від небезпечних символів. Базова модель надає уніфіковані методи для виконання запитів до бази даних: отримання списку записів, отримання одного запису або порожнього результату, виконання змін із поверненням ідентифікатора нового запису та підрахунку кількості записів за умовою. Усі ці методи використовують підготовлені запити, що забезпечує захист від підміни даних через введення користувача. Окремий клас допоміжних функцій містить інструменти загального призначення, які використовуються в різних частинах системи: перетворення тексту в безпечний формат для URL, форматування номерів телефонів, обрізання довгих текстів для анонсів, визначення діапазонів дат для

		Ступакова А.С.			ІПЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Чижомотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		41

фільтрації розкладу, а також обробка завантаження зображень із перевіркою типу та розміру файлу.

На рівні моделей зосереджена вся бізнес-логіка системи, і кожен клас відповідає за свою конкретну предметну область. Клас для роботи з користувачами керує обліковими записами всіх ролей - реєстрацією, пошуком, оновленням профілю та управлінням статусом. Він підтримує кілька сценаріїв створення облікового запису: звичайну реєстрацію через форму з електронною адресою і паролем, створення запису адміністратором вручну, а також спрощений гостьовий запис без пароля, коли людина хоче відвідати лише одне пробне заняття і пізніше може повноцінно активувати свій обліковий запис.

Найбільшим і найскладнішим є клас занять. Він охоплює вибірку та фільтрацію занять за стилем, викладачем, залом, рівнем складності та датою, логіку запису студентів із перевіркою всіх необхідних умов, фіксацію відвідуваності з автоматичним списанням заняття з абонементу, перевірку конфліктів розкладу перед створенням нового заняття - система не дозволить призначити викладача або зал на два заняття одночасно, а також нарахування рядків виплати викладачу одразу після відмітки присутності студента. Окремо підтримується створення серій регулярних занять: адміністратор обирає дні тижня та кількість тижнів, і система автоматично генерує всі заняття серії, пропускаючи ті дати, де виникає конфлікт.

Клас абонементів реалізує повний їх життєвий цикл. Якщо студент купує новий абонемент, коли попередній ще активний, новий ставиться в чергу і автоматично активується після вичерпання поточного. Система також відстежує закінчення терміну дії абонементів і автоматично змінює їх статус при наступному зверненні. Клас платежів обробляє підтвердження від платіжної системи після успішної оплати, перевіряє цифровий підпис повідомлення щоб переконатися в його автентичності, та активує відповідний абонемент. Механізм побудований так, щоб повторне отримання того самого підтвердження не призвело до подвійної активації - система перевіряє поточний статус платежу перед будь-якими змінами. Окремі класи відповідають за відгуки студентів про викладачів із перевіркою факту

		Ступакова А.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		42

відвідування, за програму лояльності з накопиченням балів і їх списанням при отриманні безкоштовного заняття, за розрахунок і виплату винагороди викладачам, а також за довідники стилів танців, залів і матеріали блогу з коментарями.

На рівні контролерів кожен клас відповідає за певний розділ системи і містить методи для обробки конкретних запитів. Контролер авторизації керує входом із захистом від перебору паролів, реєстрацією та виходом із системи з повним очищенням сесії. Контролер особистого кабінету відображає різний вміст залежно від ролі: студент бачить свою підписку, найближчі заняття і баланс балів лояльності, тоді як викладач бачить свій розклад у вигляді календаря, статистику занять і поточний баланс виплат. Окремі контролери обробляють оформлення і підтвердження оплати абонементів, публічний розклад занять із можливістю фільтрації, роботу з відгуками та статтями блогу. Найбільшим є контролер адміністратора - він охоплює весь функціонал керування системою: від управління студентами і викладачами до формування пакетних виплат і модерації відгуків, від створення занять і серій до перегляду повної фінансової статистики.

Окремо від рівнів MVC в системі реалізовано два сервісні класи, які не належать до жодного з рівнів архітектури, а виступають як спеціалізовані інструменти, що викликаються контролерами у потрібний момент. Перший відповідає за інтеграцію з платіжною системою: він формує підписані дані для передачі на сторінку оплати, перевіряє підпис вхідних повідомлень від платіжного сервісу та, за потреби, запитує актуальний статус платежу через API. Другий сервісний клас відповідає за надсилання електронних листів. Для відправки використовується бібліотека RHPMailer з підключенням через SMTP-протокол, а під час розробки та тестування застосовується спеціальний поштовий сервіс, який перехоплює всі листи і не відправляє їх реальним користувачам.[26] Система надсилає такі типи листів: підтвердження запису на заняття з деталями часу, залу та викладача, сповіщення про скасування заняття з інформацією про повернення заняття на абонемент, а також повідомлення від викладача з нотаткою до заняття, яке він може надіслати всім

		Ступакова А.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		43

записаним студентам. Кожен лист оформлений у єдиному стилі з брендингом студії та містить кнопку переходу до особистого кабінету або розкладу, що робить комунікацію з користувачем зручною та послідовною.

2.5. Опис реалізації ключових модулів системи

Розглянемо найбільш складні з технічної точки зору фрагменти реалізації, які потребують детального пояснення.

Одним із ключових модулів є інтеграція з платіжним сервісом LiqPay.[12] Процес оплати складається з двох частин: формування платіжної форми та обробки підтвердження від сервісу. При оформленні оплати система генерує унікальний ідентифікатор замовлення у форматі FLING-{userId}-{timestamp}, створює запис платежу зі статусом pending та формує підписані дані для передачі на форму LiqPay:

```
$orderId = 'FLING-' . $userId . '-' . time();

$payModel->create([
    'external_order_id' => $orderId,
    'student_user_id'   => $userId,
    'subscription_type_id' => $typeId,
    'amount'            => $type['price'],
]);

$data = $this->liqpay->buildData([
    'version'    => '3',
    'action'     => 'pay',
    'amount'     => $type['price'],
    'currency'   => 'UAH',
    'description'=> 'Абонемент «' . $type['name'] . '» – Fling Dance
Studio',
    'order_id'   => $orderId,
    'result_url' => APP_URL . '/payment/result?order_id=' .
urlencode($orderId),
    'server_url' => LIQPAY_SERVER_URL,
]);
```

Після завершення транзакції LiqPay надсилає підписаний POST-запит на окремий callback-ендпоінт. Система верифікує цифровий підпис, знаходить платіж за ідентифікатором замовлення і оновлює його статус. Перевірка на повторну обробку унеможливорює подвійну активацію абонементу у разі повторного надходження callback-запиту:

		Ступакова А.С.			ІПЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		44

```

public function callback(): void
{
    $rawData    = $_POST['data']      ?? '';
    $signature  = $_POST['signature'] ?? '';

    if (!$rawData || !$signature) {
        throw new \Core\HttpException(400, '', 'Bad Request');
    }
    if (!$this->liqpay->verifySignature($rawData, $signature)) {
        throw new \Core\HttpException(403, '', 'Invalid signature');
    }

    $payload = $this->liqpay->decodeData($rawData);
    $orderId = $payload['order_id'] ?? '';
    $status  = $payload['status']   ?? '';
    $liqpayId = (string)($payload['payment_id'] ?? '');

    if (!$orderId) {
        throw new \Core\HttpException(422, '', 'No order_id');
    }

    $payModel = new PaymentModel();
    $payment  = $payModel->findById($orderId);

    if (!$payment) {
        throw new \Core\HttpException(404, '', 'Order not found');
    }

    if ($payment['status'] === 'paid') {
        throw new \Core\HttpException(200, '', 'Already processed');
    }

    if (in_array($status, ['success', 'sandbox'], true)) {
        $payModel->updateStatus($payment['id'], 'paid', $liqpayId);
        $this->activateSubscription($payment, $payModel);
    } elseif (in_array($status, ['failure', 'error', 'reversed', 'expired'],
true)) {
        $payModel->updateStatus($payment['id'], 'failed', $liqpayId);
    }

    throw new \Core\HttpException(200, '', 'OK');
}

```

Не менш важливим є метод фіксації відвідуваності, який є найскладнішим у системі, оскільки одна операція запускає ланцюжок взаємопов'язаних дій. При відмітці присутності студента система послідовно вносить або оновлює запис у таблиці відвідуваності, визначає джерело списання, списує заняття з абонементу та нараховує виплату викладачу. При цьому логіка розгалужується залежно від типу запису: для пробного заняття

		Ступакова А.С.			ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Чижмоторя О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		45

абонемент не списується, але виплата викладачу нараховується; якщо студент відсутній з поважної причини - заняття також не списується:

```
public function saveAttendance(int $enrollmentId, string $status, int $markedBy):
void
{
    $this->ensureSettledColumn();
    $this->ensureIsTrialColumn();

    $this->execute(
        "INSERT INTO attendance (enrollment_id, status, marked_by, marked_at)
        VALUES (?, ?, ?, NOW())
        ON DUPLICATE KEY UPDATE status=VALUES(status),
marked_by=VALUES(marked_by), marked_at=NOW()",
        [$enrollmentId, $status, $markedBy]
    );

    $enrCheck = $this->queryOne(
        "SELECT settled, is_trial FROM class_enrollments WHERE id=?",
        [$enrollmentId]
    );
    if (!$enrCheck) return;
    if ($enrCheck['is_trial']) {
        $this->execute("UPDATE class_enrollments SET settled=1 WHERE id=? AND
settled=0", [$enrollmentId]);
        $this->accrueTeacherPayout($enrollmentId, null);
        return;
    }

    if ($status === 'excused') {
        $this->execute(
            "UPDATE class_enrollments SET settled=1 WHERE id=? AND
settled=0",
            [$enrollmentId]
        );
        return;
    }

    $enr = $this->queryOne(
        "SELECT settled, student_user_id, subscription_id FROM
class_enrollments WHERE id=? AND settled=0",
        [$enrollmentId]
    );
    if (!$enr) return;

    $studentId = (int)$enr['student_user_id'];
    $subId      = $enr['subscription_id'] ? (int)$enr['subscription_id'] :
null;

    if (!$subId) {
```

		Ступакова А.С.			ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		46

```

        $activeSub = $this->queryOne(
            "SELECT id FROM subscriptions
            WHERE student_user_id=? AND status='active' AND expires_at>? AND
classes_remaining>0
            ORDER BY created_at DESC LIMIT 1",
            [$studentId, date('Y-m-d H:i:s')]
        );
        $subId = $activeSub ? (int)$activeSub['id'] : null;
    }

    if ($subId) {
        $this->execute(
            "UPDATE subscriptions
            SET classes_remaining=GREATEST(CAST(classes_remaining AS
SIGNED)-1,0),
            status=IF(classes_remaining<=1,'exhausted',status)
            WHERE id=? AND classes_remaining < ?",
            [$subId, UNLIMITED_CLASSES]
        );
        $this->execute(
            "UPDATE class_enrollments SET settled=1, subscription_id=? WHERE
id=?",
            [$subId, $enrollmentId]
        );
    } else {
        $this->execute(
            "UPDATE class_enrollments SET settled=1 WHERE id=?",
            [$enrollmentId]
        );
    }

    $this->accrueTeacherPayout($enrollmentId, $subId);
}

```

Логіка черги абонементів реалізована таким чином, що при купівлі нового абонементу система перевіряє наявність вже активного. Якщо активний абонемент існує, новий отримує статус pending і ставиться в чергу. Після того як поточний абонемент вичерпується, система автоматично знаходить перший у черзі, розраховує дату завершення від поточного моменту і активує його:

```

if ($hasActive) {
    return $this->execute(
        "INSERT INTO subscriptions
        (student_user_id, subscription_type_id, classes_total,
classes_remaining,
        status, created_at, updated_at)

```

		Ступакова А.С.			ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		47

```

VALUES (?, ?, ?, ?, 'pending', NOW(), NOW())",
[$studentId, $typeId, $total, $total]
);
}

$expiresAt = date('Y-m-d H:i:s', strtotime('+ ' .
(int)$type['validity_days'] . ' days'));
return $this->execute(
    "INSERT INTO subscriptions
    (student_user_id, subscription_type_id, classes_total,
classes_remaining,
    expires_at, status, activated_at, created_at, updated_at)
    VALUES (?, ?, ?, ?, ?, 'active', NOW(), NOW(), NOW())",
    [$studentId, $typeId, $total, $total, $expiresAt]
);

```

Для надсилання email-сповіщень реалізовано окремий сервісний клас, який знаходиться поза рівнями MVC і викликається контролерами у потрібний момент. Він використовує бібліотеку PHPMailer з підключенням до поштового сервера через SMTP та підтримує п'ять типів листів: підтвердження запису на заняття, сповіщення про скасування, квитанція після оплати, повідомлення від викладача та нагадування про майбутнє заняття. Під час розробки та тестування застосовується сервіс Mailtrap, який перехоплює всі вихідні листи без реального відправлення адресатам.[26] Усі листи формуються в єдиному HTML-шаблоні з брендингом студії:

```

$mail->isSMTP();
$mail->Host      = $host;
$mail->SMTPAuth  = true;
$mail->Username  = $user;
$mail->Password  = defined('MAIL_SMTP_PASS') ? MAIL_SMTP_PASS : '';
$mail->SMTPSecure =
\PHPMailer\PHPMailer\PHPMailer::ENCRYPTION_STARTTLS;
$mail->Port      = defined('MAIL_SMTP_PORT') ? (int)MAIL_SMTP_PORT :
587;
$mail->CharSet    = 'UTF-8';

$from      = defined('MAIL_FROM')      ? MAIL_FROM      :
'noreply@fling.studio';
$fromName  = defined('MAIL_FROM_NAME') ? MAIL_FROM_NAME : 'Fling Dance
Studio';
$mail->setFrom($from, $fromName);
$mail->addAddress($to);

$mail->isHTML(true);
$mail->Subject = $subject;

```

		Ступакова А.С.			ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		48

```
$mail->Body = self::wrap($body, $subject);
```

```
return $mail->send();
```

Таким чином, розглянуті фрагменти демонструють підхід до реалізації найбільш відповідальних операцій системи, де кожна дія супроводжується перевітками, захистом від повторного виконання та автоматичними тригерами суміжних процесів.

Висновки до другого розділу

У другому розділі було спроектовано та описано структуру вебплатформи танцювальної студії - від організації бази даних до програмної реалізації окремих модулів.

На початку розділу визначено вимоги до системи та описано варіанти використання для кожної ролі: студента, викладача та адміністратора. Це дало чітке розуміння того, який функціонал має бути реалізований і для кого.

Для зберігання даних спроектовано реляційну базу даних із вісімнадцяти таблиць, що охоплюють усі процеси студії. База даних нормалізована до третьої нормальної форми, завдяки чому вдалося уникнути дублювання даних і забезпечити їх узгодженість. Цілісність підтримується на кількох рівнях одночасно - через зовнішні ключі, унікальні індекси та програмні перевірки перед виконанням операцій.

Описано алгоритми основних процесів: оплати абонементів через LiqPay, запису на заняття, обліку відвідуваності з автоматичним списанням занять і нарахуванням виплат викладачам, роботи програми лояльності та надсилання email-сповіщень користувачам.

Систему реалізовано за архітектурним шаблоном MVC, де кожен рівень виконує свою роль: моделі містять бізнес-логіку, контролери обробляють запити, а шаблони формують інтерфейс. Окремо від цих рівнів функціонує сервіс надсилання пошти, який автоматично відправляє листи при ключових подіях у системі.

		Ступакова А.С.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		49

Найскладніші частини реалізації описано з наведенням фрагментів коду та діаграмою послідовності для процесу оплати, що дозволяє наочно побачити взаємодію між компонентами системи та зовнішнім платіжним сервісом.

		Ступакова А.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		50

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБПЛАТФОРМОЮ ТАНЦЮВАЛЬНОЇ СТУДІЇ

3.1. Порядок встановлення та налаштування системи

Встановлення та налаштування вебплатформи танцювальної студії вимагає розгортання всіх її компонентів та налаштування як кожного елементу окремо, так і загальної системи в цілому. Для опису архітектури розгортання було створено діаграму розгортання (Рис. 3.1).[5]

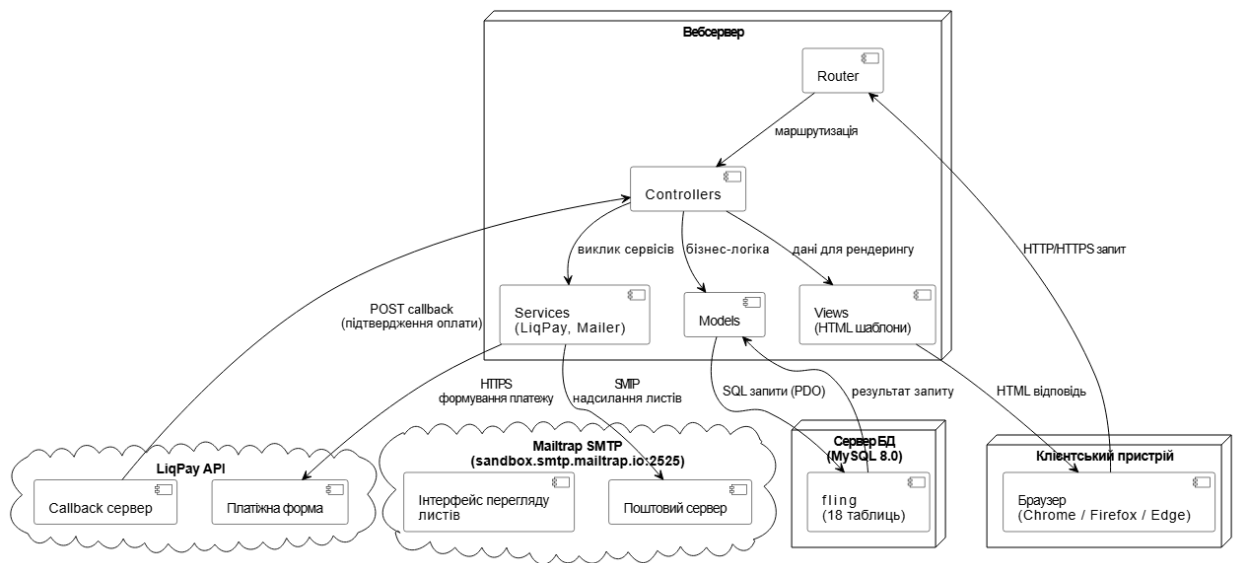


Рис. 3.1 - Діаграма розгортання

Діаграма відображає взаємодію між чотирма основними вузлами системи.

Клієнтський пристрій — будь-який комп'ютер або мобільний пристрій користувача з сучасним браузером. Через браузер надсилаються HTTP/HTTPS запити до вебсервера та отримуються готові HTML-сторінки у відповідь. Жодні дані системи не зберігаються на стороні клієнта, окрім сесійного cookie.

Вебсервер на базі Apache 2.4 і PHP 8.2 є основним вузлом системи, на якому розгорнуто весь програмний комплекс. Компонент Router приймає вхідні запити, визначає відповідний контролер і метод та перевіряє CSRF-токен для POST-запитів. Контролери обробляють запити користувачів, звертаються до моделей та передають дані в шаблони. Моделі реалізують бізнес-логіку та виконують запити до бази даних через PDO. Шаблони

формують HTML-відповідь на основі даних отриманих від контролера. Окремо від рівнів MVC функціонують сервісні класи для роботи з LiqPay та надсилання email-сповіщень через PHPMailer.

Сервер бази даних MySQL 8.0 зберігає всі дані системи в базі fling, яка містить вісімнадцять таблиць. Взаємодія з базою даних відбувається виключно через PDO з підготовленими запитами, що унеможливорює SQL-ін'єкції.

LiqPay API є зовнішнім платіжним сервісом. Взаємодія відбувається у двох напрямках: система формує підписані дані та перенаправляє користувача на платіжну форму LiqPay, а після завершення транзакції LiqPay надсилає підписаний POST callback-запит назад на сервер для підтвердження оплати.

Mailtrap SMTP є поштовим сервісом, що використовується під час розробки та тестування. Він перехоплює всі вихідні листи та відображає їх у зручному веб-інтерфейсі без реального відправлення адресатам.

Для того щоб система запрацювала, потрібно підготувати серверне середовище. Обов'язковими компонентами є PHP версії 8.2, MySQL 8.0 для зберігання даних, вебсервер Apache з увімкненим модулем mod_rewrite, менеджер залежностей Composer та бібліотека PHPMailer для надсилання листів.

Встановлення відбувається покроково. Спочатку файли проекту копіюються на сервер, після чого виконується команда `composer install` яка автоматично підтягує всі необхідні бібліотеки. Далі створюється база даних і імпортується дамп із готовою структурою таблиць. Після цього налаштовуються конфігураційні файли - прописуються дані для підключення до бази даних, ключі платіжного сервісу та параметри пошти. На завершення перевіряється наявність файлу `.htaccess` та надаються права на запис для директорії `uploads/`, де зберігаються завантажені зображення.

Поведінку системи можна гнучко налаштовувати через параметри у файлі конфігурації. Зокрема, визначається скільки часу триває сесія користувача, за скільки годин до заняття студент може скасувати запис безкоштовно, скільки балів нараховується за одне відвідане заняття та скільки потрібно накопичити для отримання безкоштовного заняття. Також тут

		Ступакова А.С.			ІПЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		52

вказується чи працює платіжний сервіс у тестовому режимі - під час розробки використовується пісочниця LiqPay, яка дозволяє виконувати тестові платежі без реального списання коштів.

Доступ до розділів системи розмежовано за трьома ролями. Студент працює зі своїм особистим кабінетом - переглядає розклад, записується на заняття, оплачує абонементи та користується програмою лояльності. Викладач бачить свій розклад, відмічає присутність студентів на заняттях та переглядає нараховану винагороду. Адміністратор має повний доступ до всієї системи - керує студентами, викладачами, заняттями, платежами, виплатами та контентом. Перевірка ролі відбувається автоматично перед виконанням кожного запиту на рівні базового контролера.

Окрім локального розгортання, у проєкті налаштовано автоматичний конвеєр CI/CD на платформі GitLab. Щоразу коли до репозиторію потрапляють нові зміни, система автоматично встановлює залежності, запускає тести та збирає Docker-образ застосунку.[20] Це дозволяє одразу виявляти помилки після кожної зміни коду, не чекаючи ручної перевірки. Розгортання на основну гілку відбувається лише після того як всі попередні етапи завершилися успішно.[4, 19, 23]

3.2. Інтерфейс користувача

Інтерфейс вебплатформи танцювальної студії розроблено з урахуванням потреб трьох різних типів користувачів - студентів, викладачів та адміністратора. Кожна роль має власний набір сторінок та функціональних можливостей. Публічна частина сайту доступна без авторизації та орієнтована на залучення нових клієнтів, тоді як особисті кабінети надають авторизованим користувачам доступ до відповідного функціоналу.

Головна сторінка є точкою входу для нових відвідувачів. На ній розміщено короткий опис студії, найближчі заняття з розкладу, перелік танцювальних напрямків та кнопки для переходу до розкладу і реєстрації (Рис. 3.2).

		Ступакова А.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		53

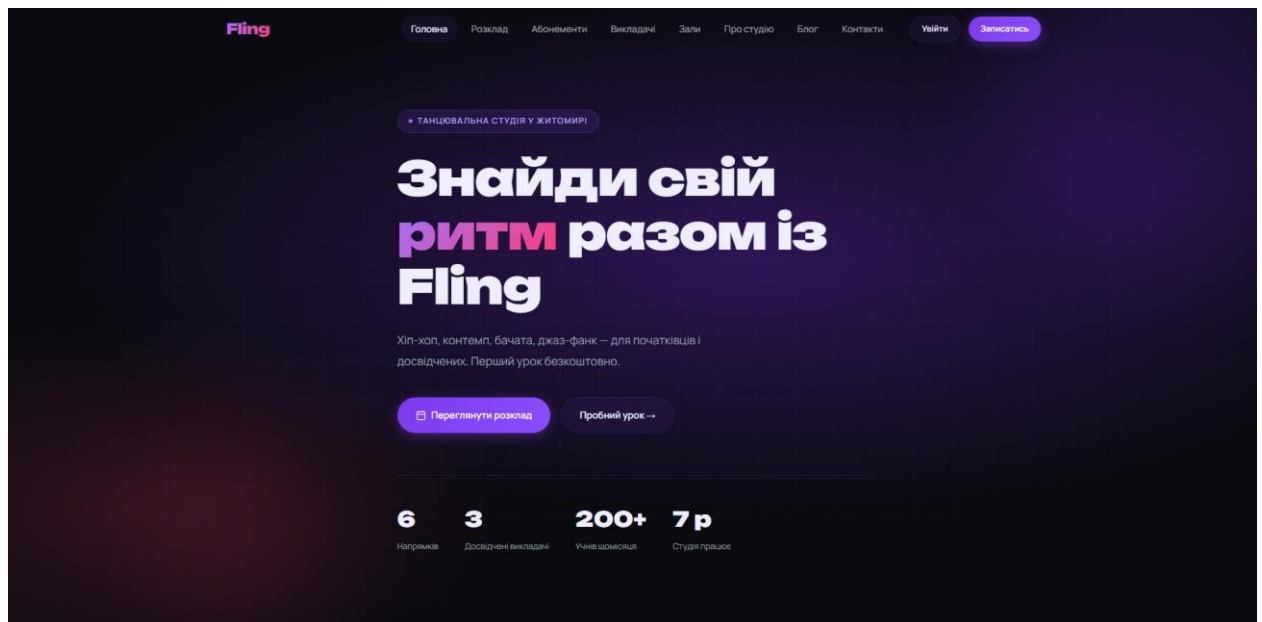


Рис. 3.2 - Головна сторінка

Сторінка розкладу відображає всі заплановані заняття у вигляді карток. Користувач може фільтрувати заняття за стилем танцю, викладачем, рівнем складності, залом та датою. Кожна картка заняття містить назву, час проведення, зал, викладача та кількість вільних місць. Авторизований студент з активним абонементом може одразу записатися на заняття (Рис. 3.3).

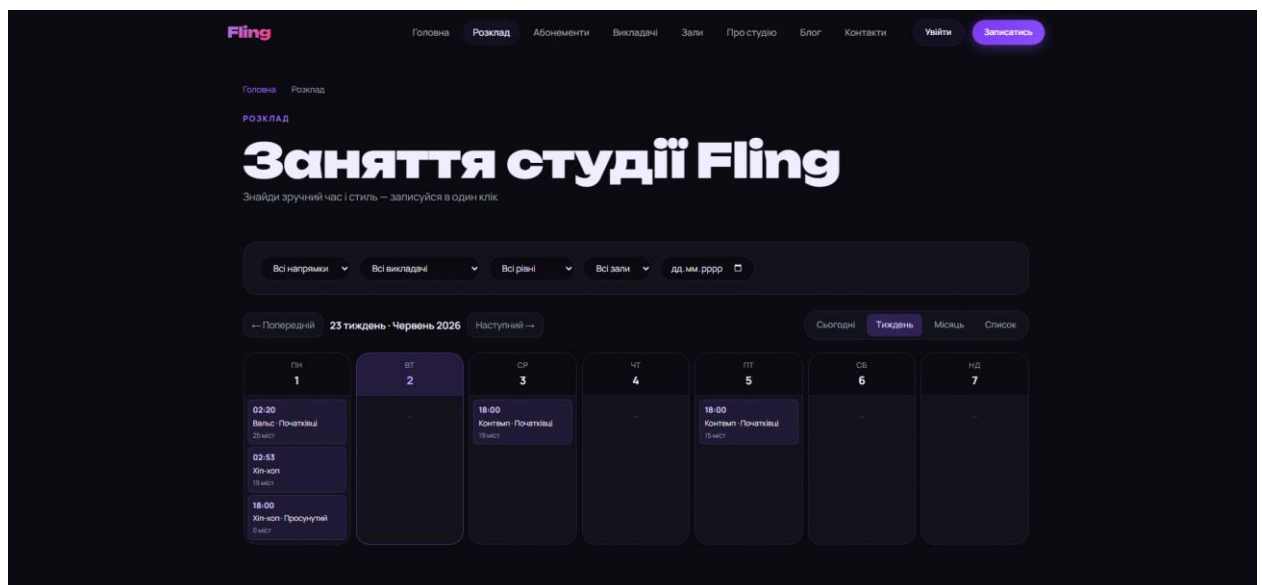


Рис. 3.3 - Сторінка розкладу

Сторінка викладачів відображає публічні профілі всіх активних викладачів студії з фотографією, біографією, стилями танців та середнім рейтингом на основі відгуків студентів. При переході на сторінку конкретного викладача відображається детальна інформація та відгуки студентів. Студент

		Ступакова А.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		54

може залишити відгук після підтвердженого відвідування заняття цього викладача (Рис. 3.4).

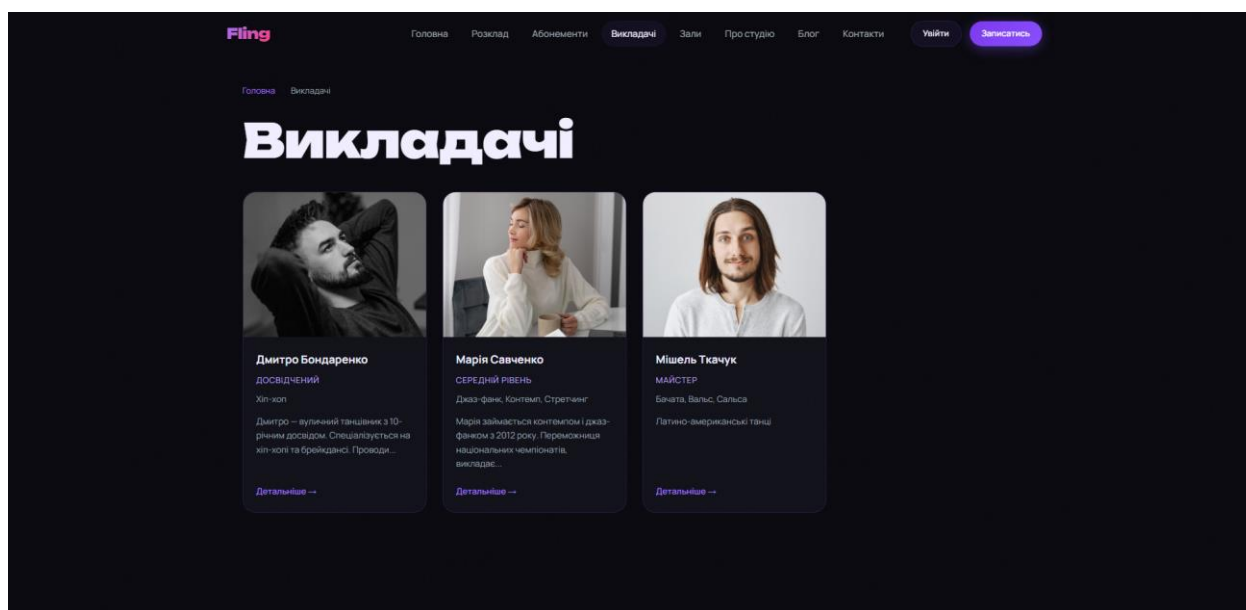


Рис. 3.4 - Сторінка викладачів

Розділ блогу містить статті та новини студії. Кожна публікація має заголовок, обкладинку, короткий анонс та повний текст. Авторизовані користувачі можуть залишати коментарі до статей (Рис. 3.5).

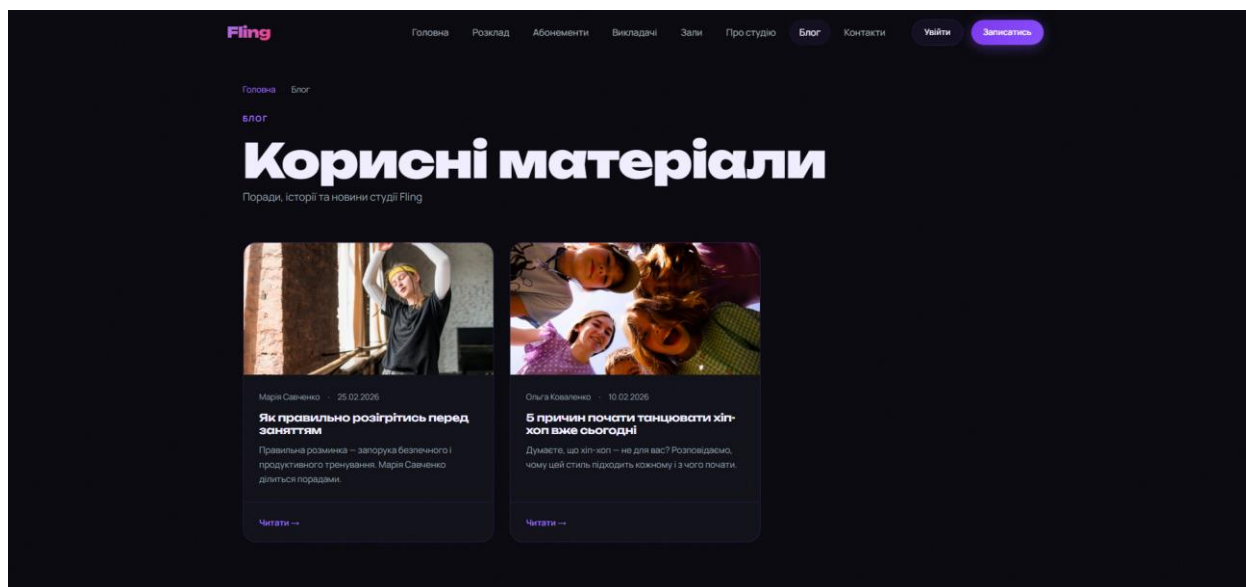


Рис. 3.5 - Сторінка блогу

Сторінка авторизації містить форму входу з полями для електронної пошти та пароля. Якщо користувач ввів невірні дані більше п'яти разів поспіль з однієї IP-адреси, система тимчасово блокує подальші спроби входу (Рис. 3.6).

		Ступакова А.С.			ІПЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		55

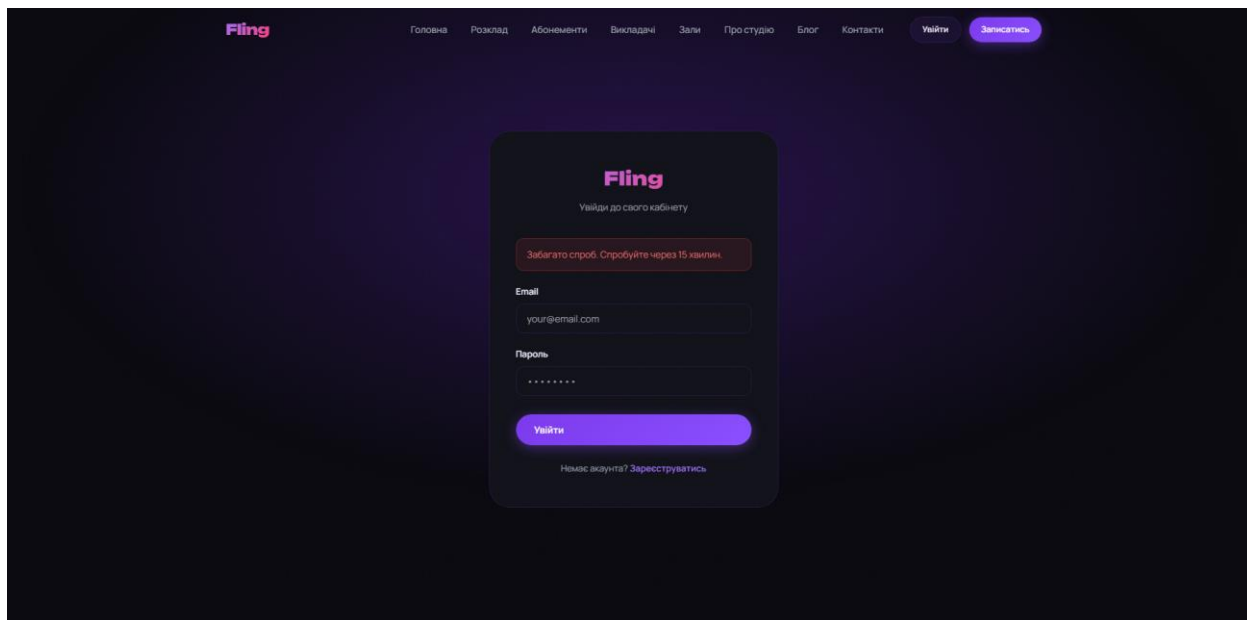


Рис. 3.6 - Сторінка входу

Реєстрація нового студента відбувається через форму з полями для імені, прізвища, електронної пошти, номера телефону та пароля. Після успішної реєстрації користувач автоматично авторизується і потрапляє до особистого кабінету (Рис. 3.7).

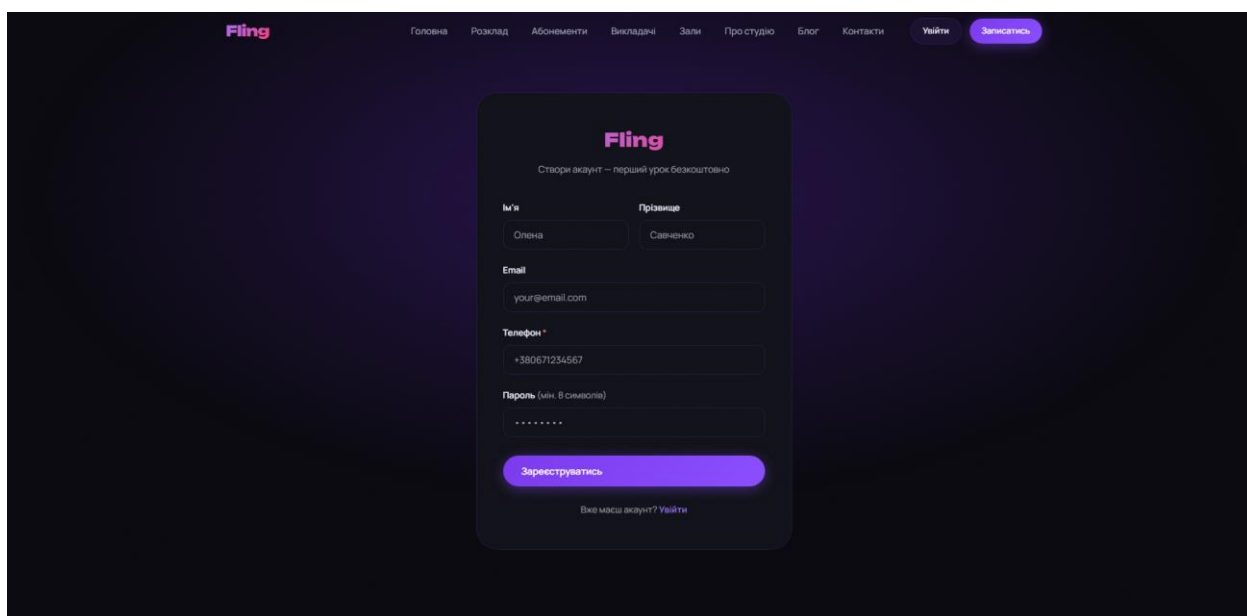


Рис. 3.7 - Сторінка реєстрації

Після авторизації студент потрапляє на дашборд особистого кабінету. На ньому відображається стан активного абонементу з кількістю занять що залишились і датою завершення, найближчі записані заняття та поточний баланс балів програми лояльності (Рис. 3.8).

		Ступакова А.С.			ІПЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		56

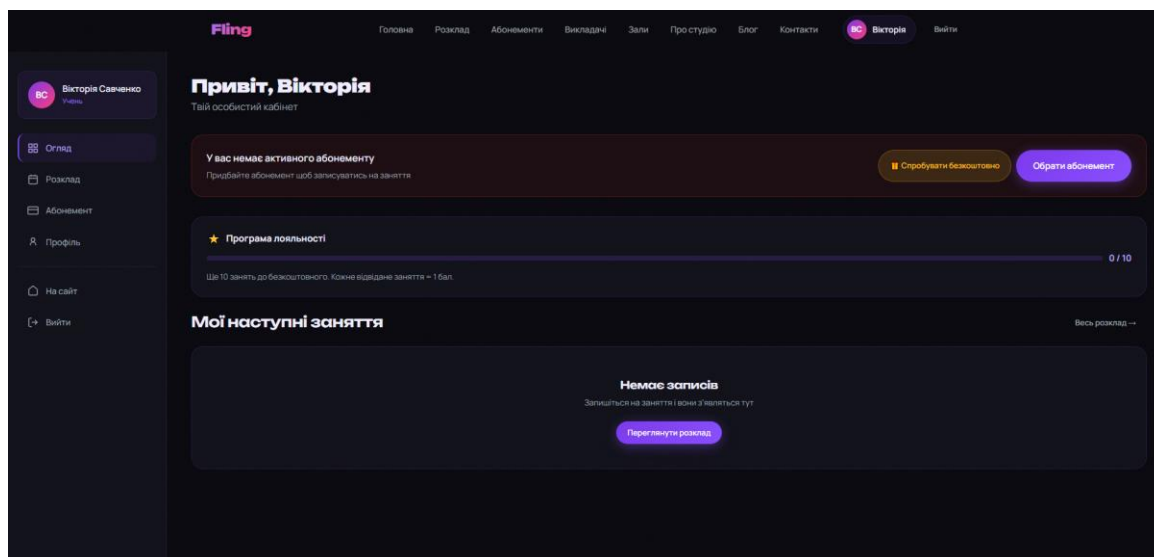


Рис. 3.8 - Сторінка кабінету студента

Розділ розкладу в кабінеті студента відображає всі заняття на які він записаний, а також дозволяє переглядати доступні заняття та записуватися на нові. При записі система автоматично перевіряє наявність активного абонементу та відсутність накладань у розкладі. Після успішного запису студент отримує підтвердження на електронну пошту (Рис. 3.8).

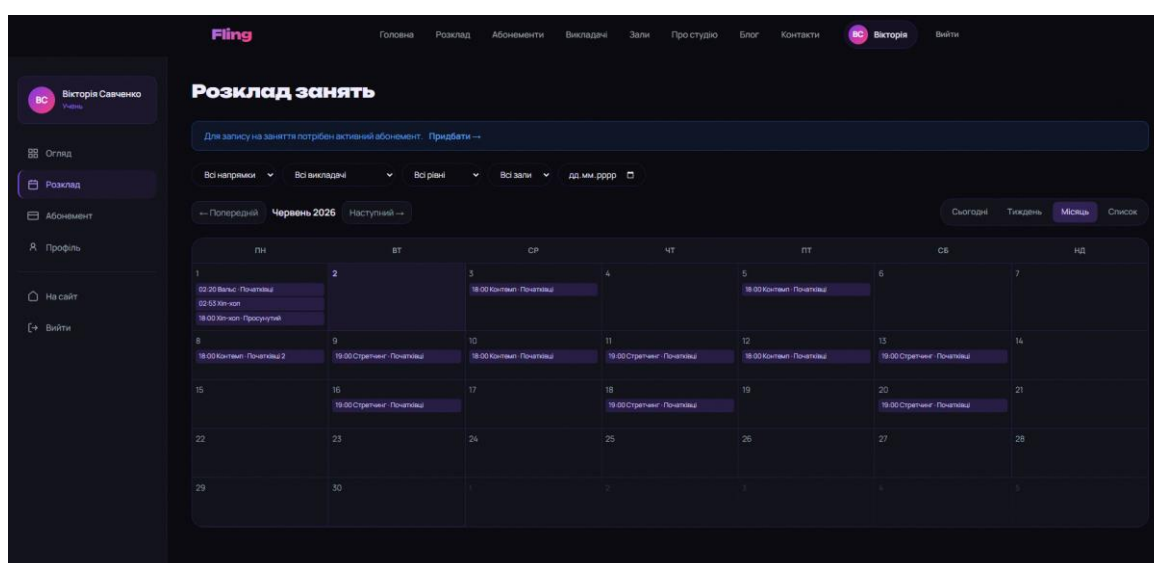


Рис. 3.9 - Сторінка розділу розклад

Розділ абонементів відображає поточний активний абонемент, його тип, кількість занять що залишилась і дату завершення, а також список попередніх абонементів. Тут же студент може придбати новий абонемент - при наявному активному він стане в чергу і активується автоматично (Рис. 3.10).

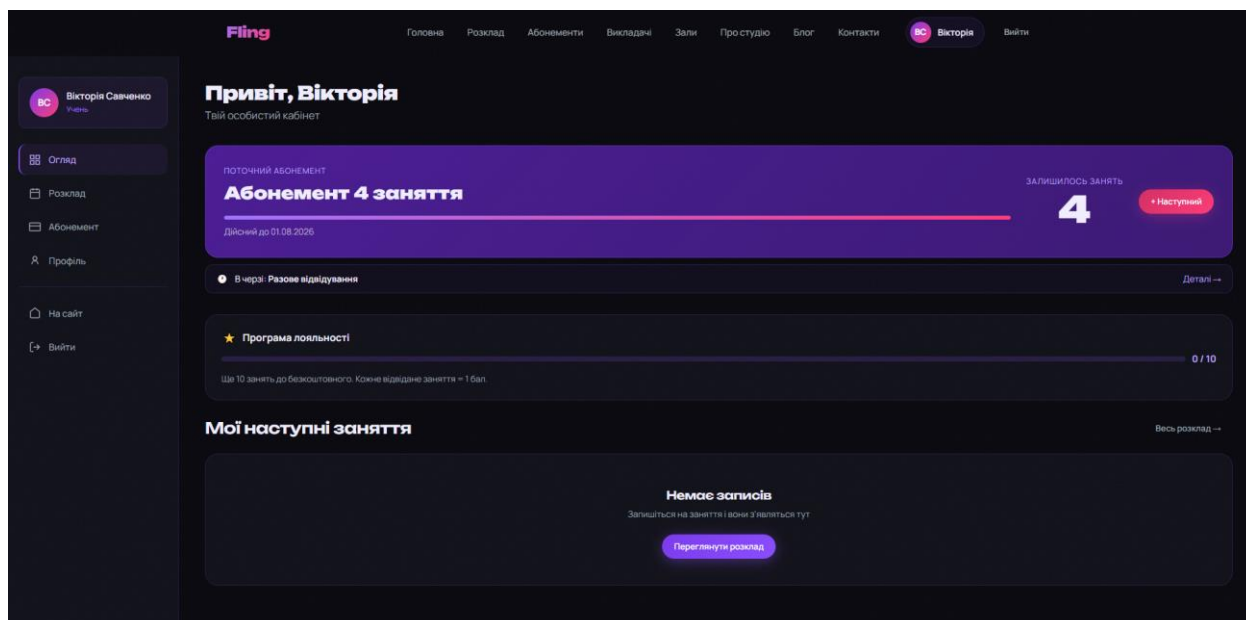


Рис. 3.10 - Сторінка розділу абонементи

Профіль користувача містить персональну інформацію студента, зокрема ім'я, прізвище, адресу електронної пошти, номер телефону. У цьому розділі користувач може редагувати власні контактні дані, змінювати пароль доступу до облікового запису (Рис. 3.11).

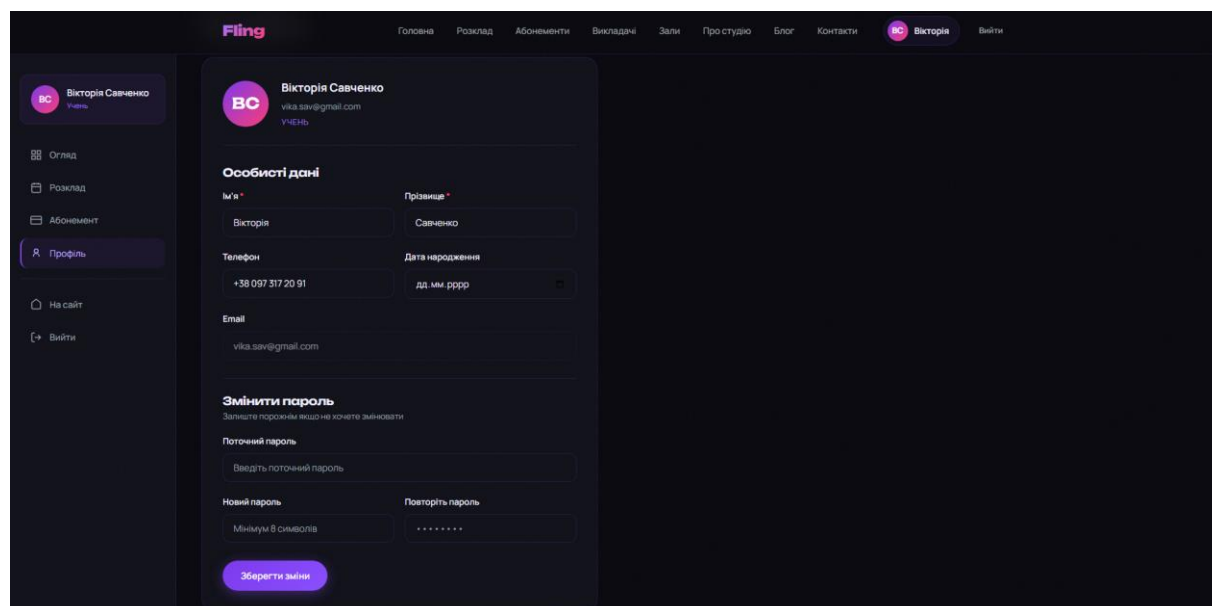


Рис. 3.11 - Сторінка профілю

Кабінет викладача містить розклад власних занять із можливістю фільтрації за датою, стилем та залом. Для кожного заняття відображається кількість записаних студентів та статус. Викладач може переглянути список студентів записаних на конкретне заняття та додати нотатку яка автоматично надійде студентам на пошту, а також баланс на рахунку (Рис. 3.12).

		Ступакова А.С.			ІПЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		58

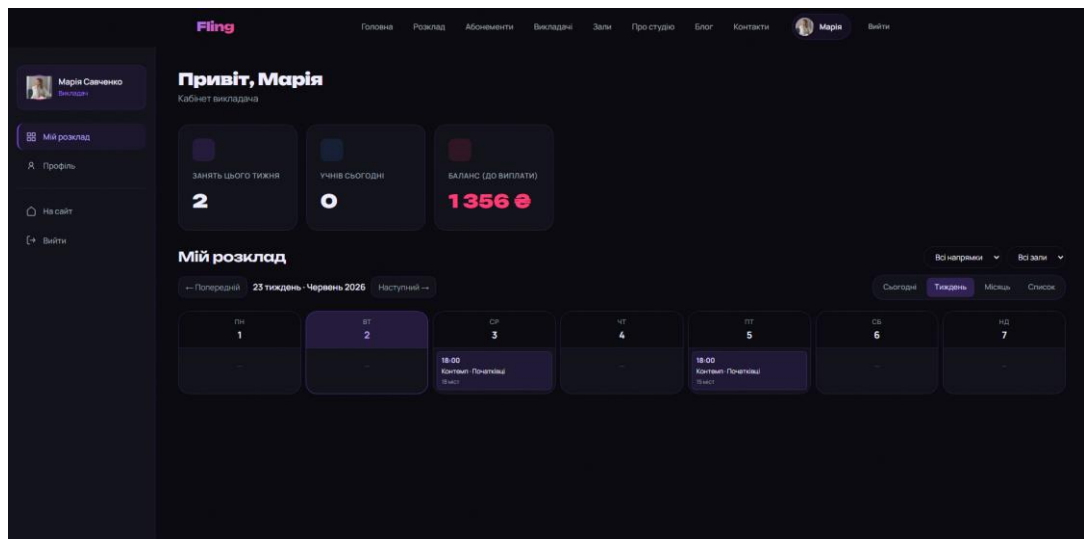


Рис. 3.12 - Сторінка кабінету викладача

Розділ відмітки відвідуваності відображає список студентів записаних на заняття з можливістю відмітити кожного як присутнього, відсутнього або відсутнього з поважної причини. Після збереження відмітки система автоматично виконує ряд операцій - списання занять з абонементу та нарахування виплати викладачу (Рис. 3.13).

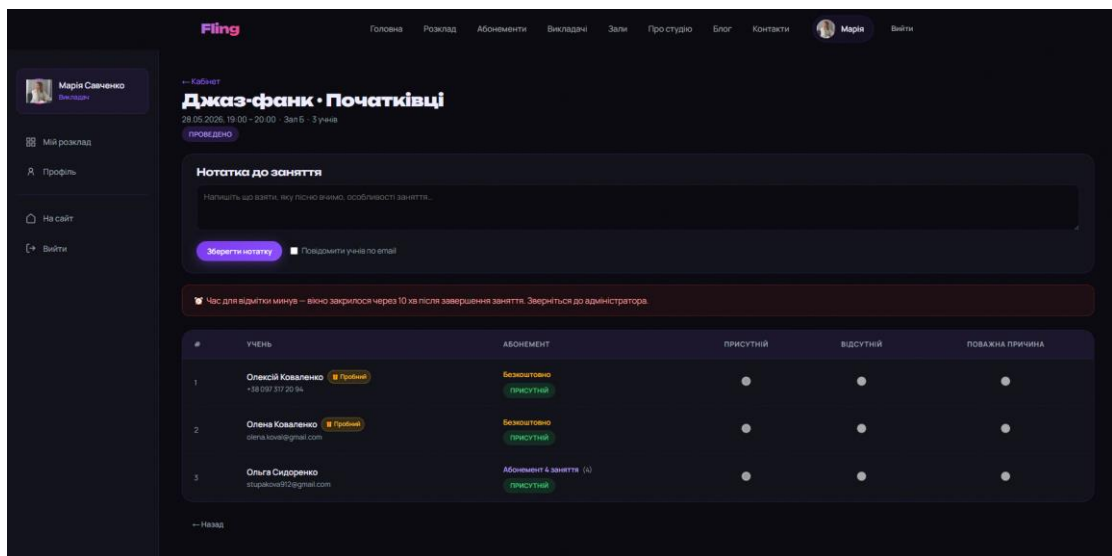


Рис. 3.13 - Сторінка відвідуваності

Адміністративна панель має власний макет з окремою навігацією та доступна виключно користувачам з роллю адміністратора. Дашборд панелі відображає ключові показники роботи студії + кількість активних студентів, заняття на сьогодні та завтра, виручку за поточний місяць та абонементи, що скоро згорять (Рис. 3.14).

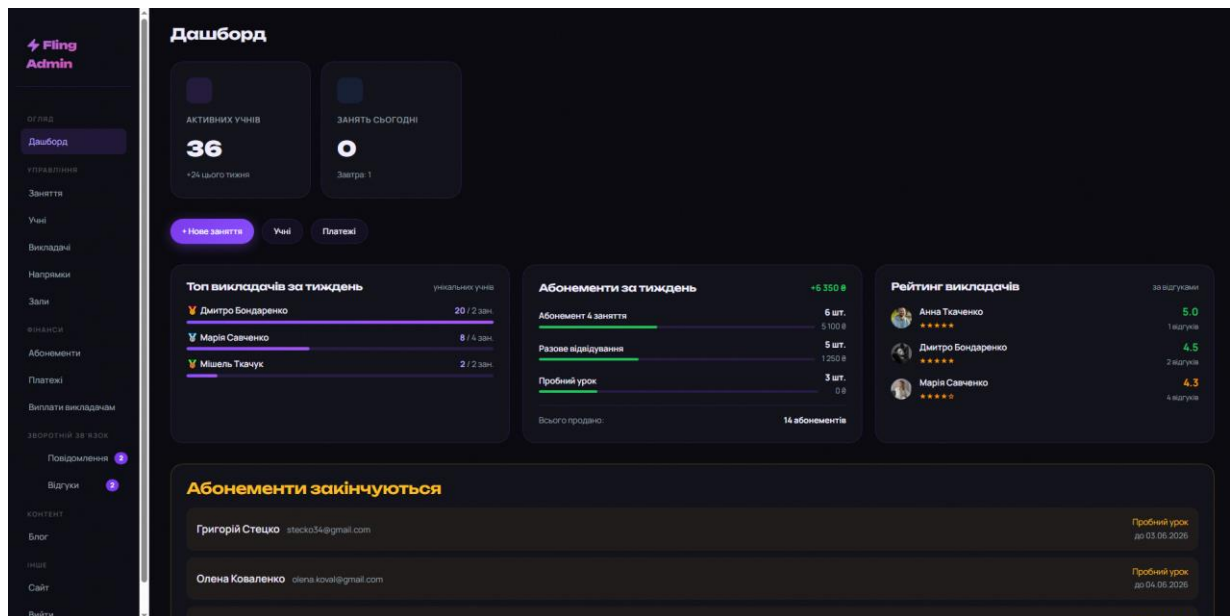


Рис. 3.14 - Сторінка адміна

Розділ управління студентами містить список усіх зареєстрованих студентів із можливістю пошуку та фільтрації. Адміністратор може переглянути детальну інформацію про кожного студента, а також заблокувати або розблокувати обліковий запис (Рис. 3.15).

Коробченко Ольга	+38 097 317 22 98	активний	1	активний	Детальніше
Олена Коваленко	+38 097 318 20 98	активний	1	активний	Детальніше
Олексій Коваленко	+38 097 317 20 94	немає	—	активний	Детальніше
Григорій Стецько	+38 097 317 20 98	активний	1	заблоковано	Детальніше
Анна Ступакова	+38 097 317 20 92	активний	1	активний	Детальніше
Ольга Сидоренко	+38 097 192 46 08	активний	4	активний	Детальніше
Іван Петренко	+38 06 76 78 90 01	активний	2	активний	Детальніше
Новий Учень	+38 06 77 89 01 01	немає	—	активний	Детальніше
Олег Заблокований	+38 06 76 78 91 01	немає	—	заблоковано	Детальніше
Юлія Киріленко	+38 06 75 67 81 01	немає	—	активний	Детальніше
Руслан Поліщук	+38 06 74 56 71 01	активний	2	активний	Детальніше
Вікторія Тимченко	+38 06 74 56 61 01	немає	—	активний	Детальніше
Андрій Литвиненко	+38 06 72 34 01 01	немає	—	активний	Детальніше

Рис. 3.15 - Сторінка управління студентами

Розділ управління викладачами дозволяє створювати нові облікові записи викладачів, редагувати їхні профілі, призначати категорії та відсоток ставки винагороди, а також керувати стилями танців (Рис. 3.16).

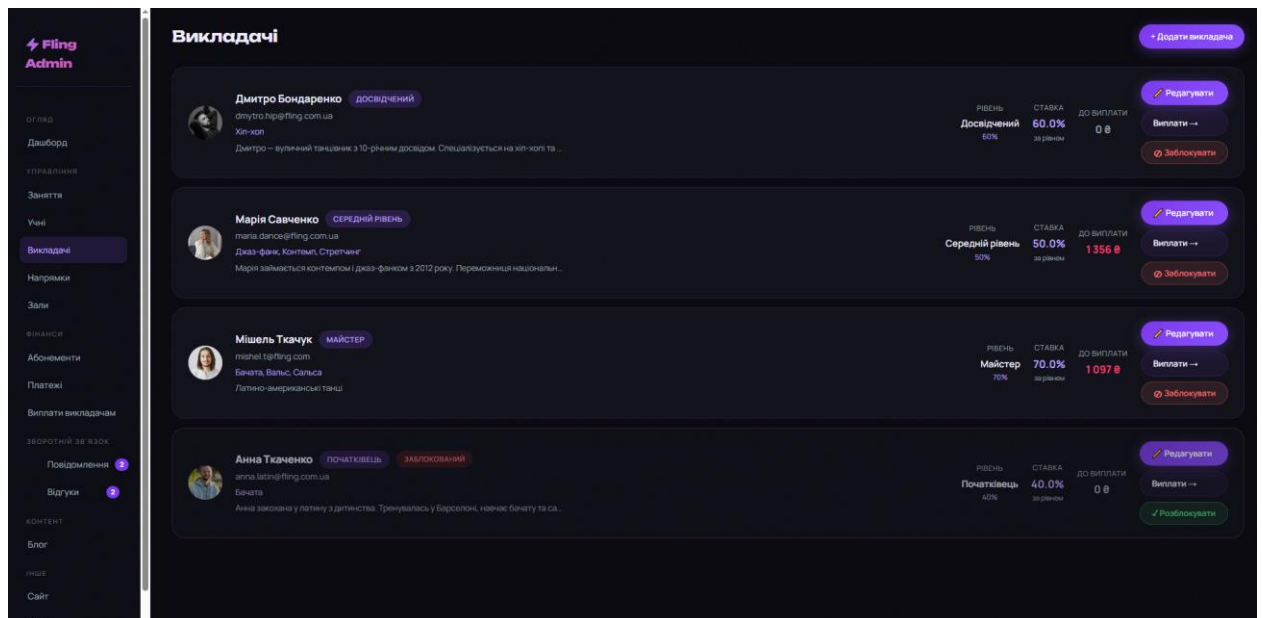


Рис. 3.16 - Сторінка управління викладачами

Розділ управління заняттями надає можливість створювати окремі заняття або серії занять що повторюються, редагувати існуючі та скасовувати їх із зазначенням причини. При створенні система автоматично перевіряє конфлікти розкладу для викладача та залу (Рис. 3.17).

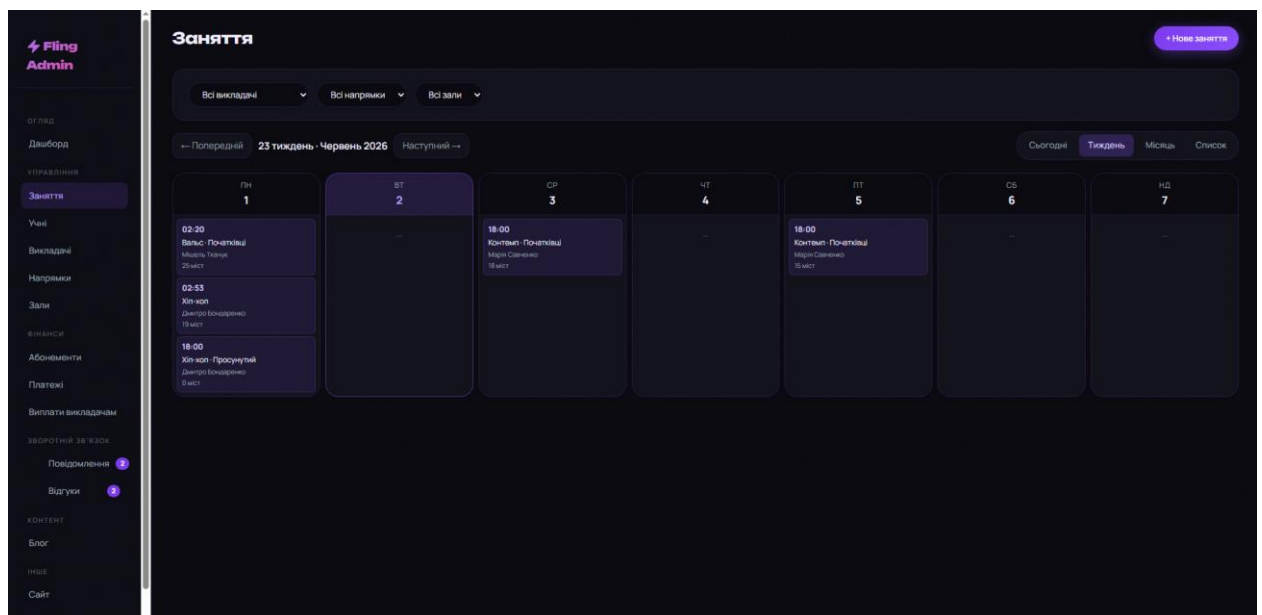


Рис. 3.17 - Сторінка управління заняттями

Розділ платежів відображає всі транзакції системи з можливістю фільтрації за статусом, методом оплати та датою. Адміністратор може також вручну створити платіж готівкою або через термінал для студентів які оплачують безпосередньо в студії (Рис. 3.18).

		Ступакова А.С.			ІПЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		61

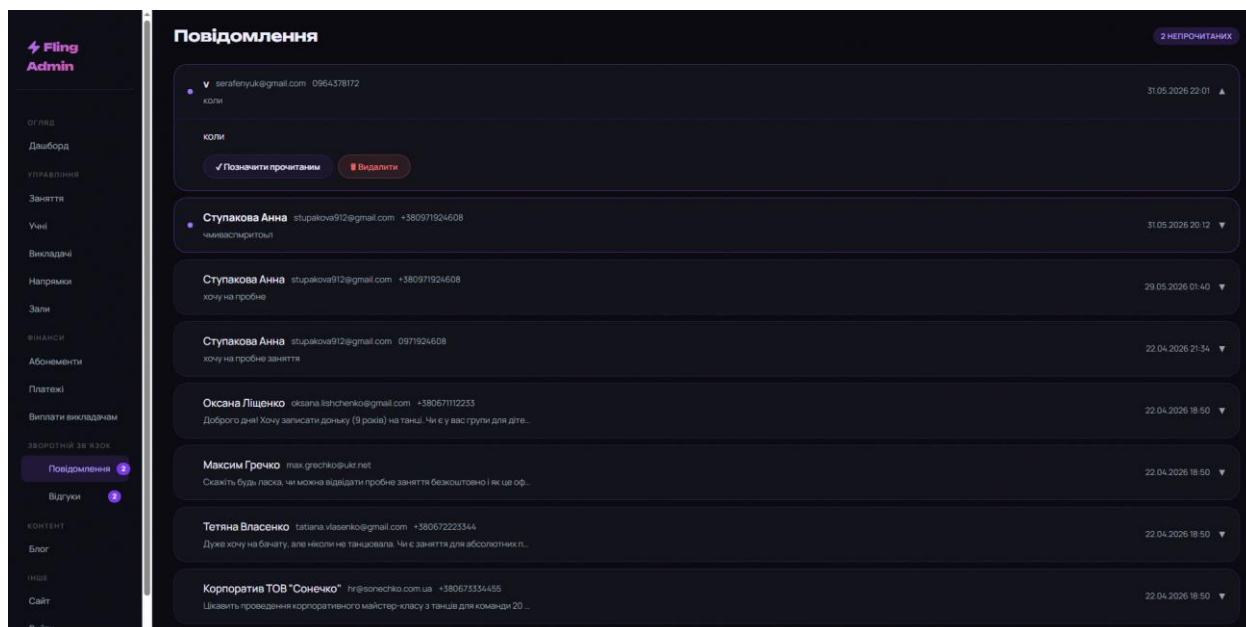


Рис. 3.20 - Сторінка повідомлень

Розділ відгуків призначений для перегляду та модерації відгуків студентів про викладачів і заняття. Адміністратор може переглядати всі залишені відгуки, а також приховувати відгуки, що порушують правила платформи. Викладачі можуть переглядати відгуки про власні заняття та аналізувати середню оцінку своєї роботи (Рис. 3.21).

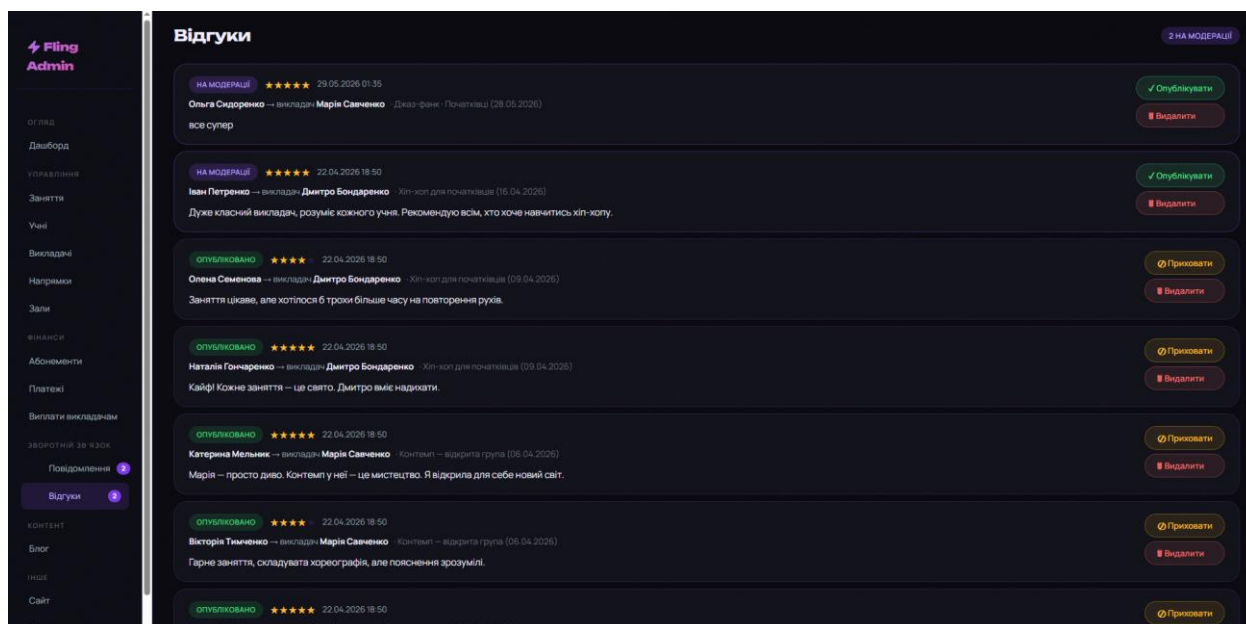


Рис. 3.21 - Сторінка відгуків

Розділ управління блогом надає адміністратору можливість створювати, редагувати, публікувати та видаляти статті й новини студії. Для кожної публікації можна вказати заголовок, короткий опис, основний текст, автора та обкладинку. Також передбачено можливість збереження чернеток (Рис. 3.22).

		Ступакова А.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		63

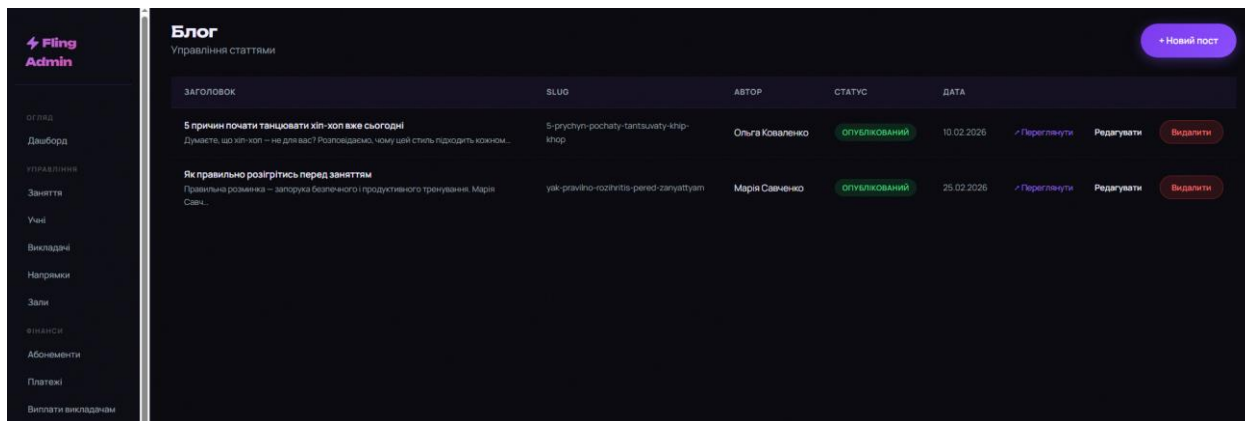


Рис. 3.22 - Сторінка управління блогом

Розділ управління танцювальними напрямками дозволяє адміністратору керувати переліком стилів танців, доступних у студії. Для кожного напрямку можна вказати назву, опис, рівень складності та пов'язаних викладачів. Інформація про напрямки використовується на публічній частині сайту та під час фільтрації занять у розкладі (Рис. 3.23).

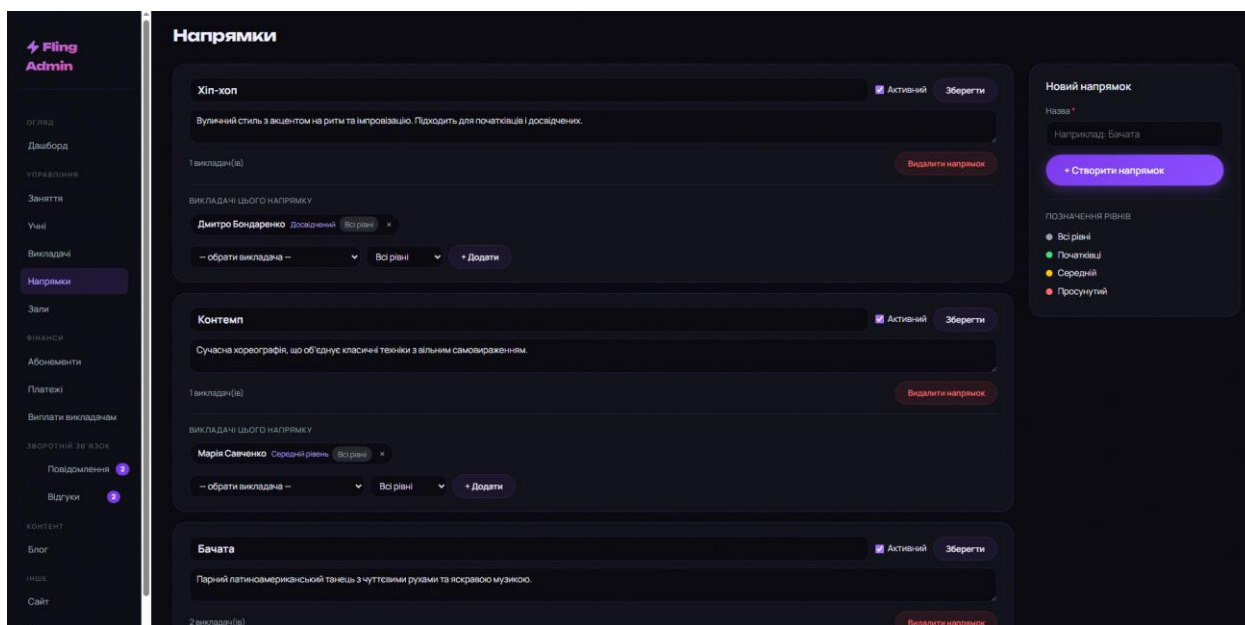


Рис. 3.23 - Сторінка управління напрямками

Сторінка оплати абонементу доступна студентам під час придбання нового абонементу або продовження існуючого. На сторінці відображаються доступні тарифні плани із зазначенням вартості, кількості занять та терміну дії. Після вибору абонементу користувач переходить до оформлення платежу. Після успішного проведення транзакції система автоматично створює абонемент, активує його або додає до черги активації, а також надсилає

підтвердження на електронну пошту. Історія всіх оплат зберігається в особистому кабінеті студента (Рис. 3.24).

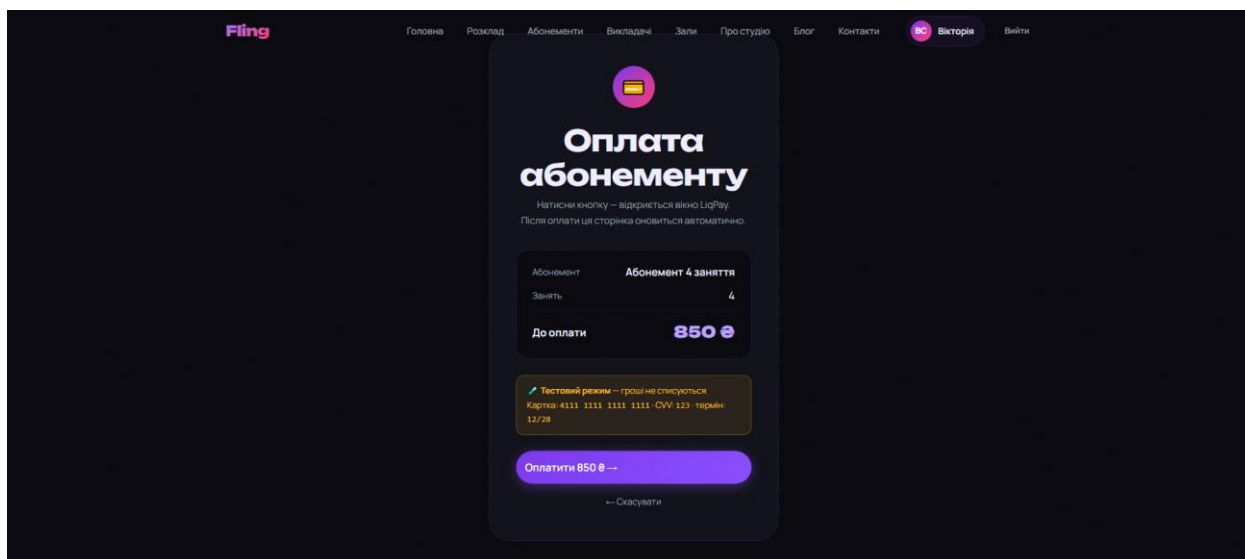


Рис. 3.24 - Сторінка оплати

Сторінка запису на заняття надає студенту можливість швидко зареєструватися на обране заняття. Під час запису відображається детальна інформація про заняття, зокрема назва, танцювальний напрямок, викладач, дата та час проведення, зал, рівень складності, а також кількість вільних місць. Перед підтвердженням запису система автоматично перевіряє наявність активного абонементу, достатню кількість доступних занять та відсутність накладань із вже запланованими заняттями користувача. Після успішного запису місце резервується за студентом, інформація про заняття додається до його особистого розкладу, а на електронну пошту надсилається лист-підтвердження із зазначенням усіх деталей заняття (Рис. 3.26). Сторінка запису на заняття наведена на Рис. 3.25.

		Ступакова А.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		65

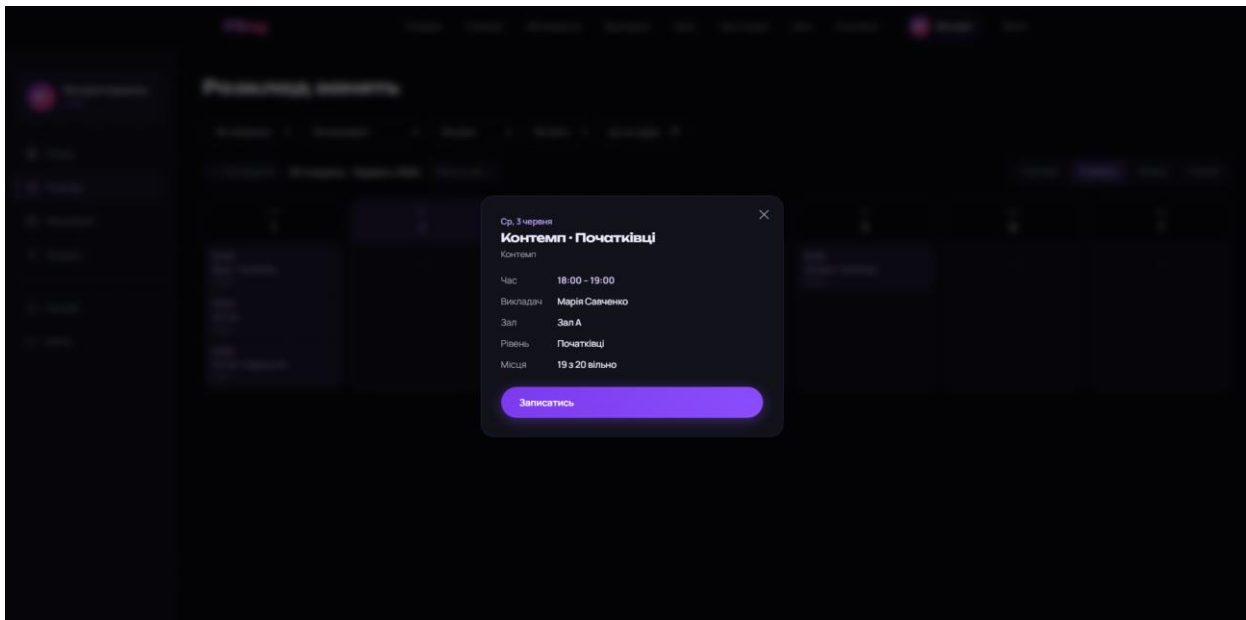


Рис. 3.25 - Сторінка запису на заняття

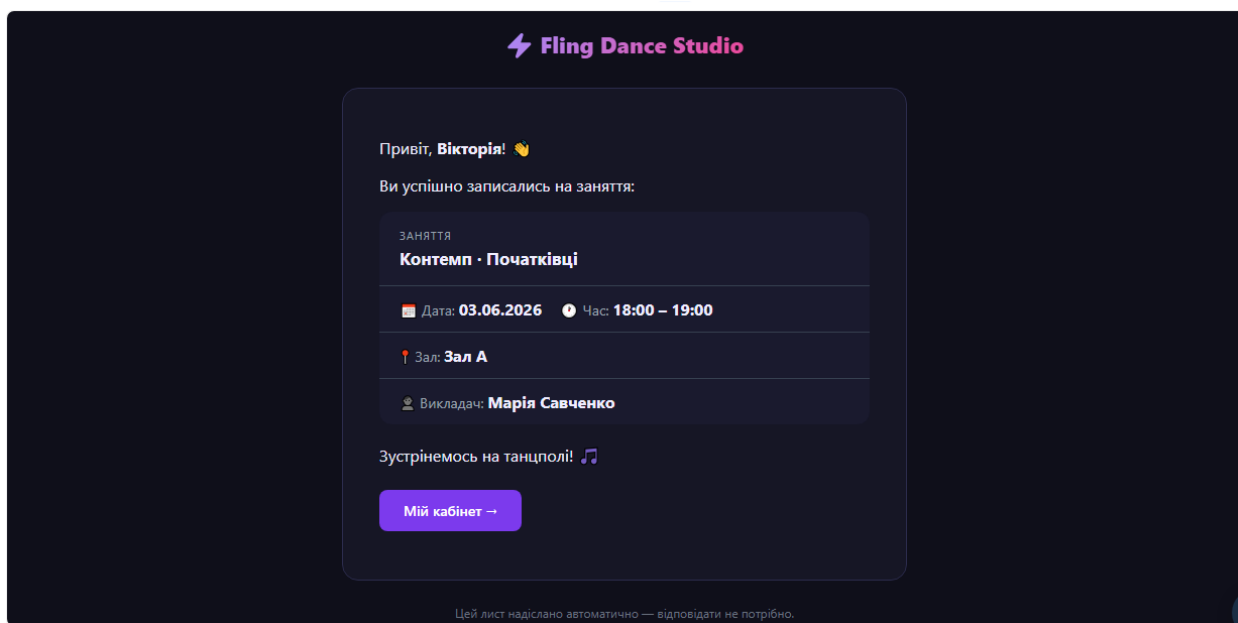


Рис. 3.26 - Лист підтвердження на пошті

Таким чином, інтерфейс користувача вебплатформи танцювальної студії спроектований з урахуванням потреб усіх категорій користувачів та забезпечує зручну й інтуїтивно зрозумілу взаємодію з системою. Чітке розмежування кабінетів за ролями, логічна структура навігації та інформативне відображення даних дозволяють клієнтам, викладачам та адміністратору ефективно виконувати свої задачі без зайвих труднощів. Реалізований інтерфейс повністю відповідає функціональним вимогам

		Ступакова А.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		66

системи та створює комфортне середовище для повсякденного використання платформи.

3.3. Тестування системи

Тестування програмного комплексу проводилось за допомогою фреймворку PHPUnit 11 у середовищі PHP 8.2. Тести організовані у три групи: модульні перевірки окремих компонентів, інтеграційні тести бізнес-логіки на рівні моделей та HTTP-тести, які імітують реальні запити до системи через маршрутизатор. Такий поділ дозволяє виявляти проблеми на різних рівнях і чітко розуміти, яка саме частина системи поводить себе некоректно.[2, 16]

Щоб тести не впливали на реальні дані, кожен тест виконується всередині транзакції бази даних, яка автоматично скасовується після його завершення. Це означає, що будь-які зміни, внесені під час тесту - нові записи, оновлення, видалення зникають без сліду після завершення перевірки і не впливають ні на наступні тести, ні на реальні дані системи. Операції зі зміни структури бази даних виконуються один раз перед початком усіх тестів і не повторюються всередині кожного тесту окремо - це важливо, оскільки такі операції в MySQL автоматично фіксують будь-яку відкриту транзакцію, що могло б порушити ізоляцію тестів і призвести до непередбачуваного збереження тестових даних.

Окрему увагу приділено тестуванню безпеки. Перевіряється, що система правильно розмежовує доступ залежно від ролі користувача: незареєстрований відвідувач не може потрапити до особистого кабінету без входу і перенаправляється на сторінку авторизації, студент не бачить адміністративних сторінок і отримує відмову з кодом 403, викладач не має доступу до панелі адміністратора навіть якщо спробує відкрити її напямуча через адресний рядок браузера. Перевіряється захист форм від підробки запитів - будь-який POST-запит без коректного CSRF-токена отримує відповідь з кодом 419 і не виконує жодних дій.[3] Єдиний виняток - зовнішній

		Ступакова А.С.			ІПЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		67

запит від платіжної системи, де замість токена використовується криптографічний підпис, що також перевіряється окремим тестом.

Окремо тестується завантаження файлів. Система відхиляє PHP-скрипти та виконувані файли, зображення з підробленим розширенням, порожні файли та файли, що перевищують допустимий розмір у п'ять мегабайт. Принципово важливо, що перевірка відбувається не за назвою файлу, яку легко підробити, а за реальним вмістом - система зчитує перші байти файлу і визначає його справжній тип. Це унеможлиблює завантаження шкідливого виконаного файлу під виглядом зображення. Також тестується стійкість пошуку до SQL-ін'єкцій: спроба передати у пошуковий рядок конструкції, що можуть змінити логіку SQL-запиту, обробляється коректно і не призводить до витоку або пошкодження даних.

Значна частина тестів охоплює бізнес-логіку роботи із заняттями та абонементом. Перевіряється, що запис студента на заняття коректно змінює свій статус при скасуванні або розрахунку і що кредити абонементу повертаються лише тоді, коли це передбачено правилами - наприклад, при скасуванні з достатнім запасом часу до початку заняття, але не при скасуванні в останній момент. Тестується каскадна поведінка при видаленні пов'язаних об'єктів: якщо адміністратор скасовує заняття або блокує викладача, система автоматично скасовує всі записи студентів на ці заняття і повертає їм відповідні кредити. Студенти при цьому отримують сповіщення, що також фіксується в тестах через перевірку виклику відповідних методів.

Перевіряється алгоритм виявлення конфліктів у розкладі. Два заняття в одній залі або з одним викладачем у той самий час не можуть існувати одночасно, і система відмовляє в створенні такого заняття. При цьому заняття, що стоять одне за одним без жодного перетину за часом, конфліктом не вважаються, що також підтверджується окремими тестами. Особливо перевіряється сценарій редагування існуючого заняття: система повинна виключати саме це заняття з перевірки на конфлікт, інакше редагування будь-якого заняття завжди видавало б конфлікт з самим собою.

		Ступакова А.С.			ІПЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Чижомотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		68

Окремо перевіряється логіка абонементів у різних сценаріях. Якщо у студента вже є активний абонемент, наступний придбаний стає в чергу та активується автоматично після вичерпання поточного - тест перевіряє, що перехід відбувається коректно і жодне заняття при цьому не втрачається. Безлімітний абонемент при жодному сценарії не зменшує лічильника занять, навіть якщо студент відвідує десятки занять поспіль. Пробне заняття можна відвідати лише один раз: повторна спроба записатися блокується, а якщо перше пробне заняття було скасовано до фіксації відвідування - право на пробне повертається і студент може скористатися ним знову. Тестується також нарахування боргу: якщо на момент фіксації відвідування у студента немає активного абонементу, система фіксує борг, і наступний запис стає недоступним до його погашення. Придбання нового абонементу автоматично закриває всі борги.

Тестування платіжної системи зосереджено на коректності криптографічного підпису та обробці різних сценаріїв підтвердження оплати. Перевіряється, що успішне підтвердження від платіжної системи активує абонемент, а невдала або скасована транзакція не призводить до жодних змін у стані підписки. Повторне отримання того самого підтвердження не спричиняє подвійної активації - система перевіряє поточний статус платежу і ігнорує дублікати. Підроблений підпис або відсутність обов'язкових полів у запиті завжди призводять до відмови без будь-яких змін у базі даних.

Програма лояльності тестується на нарахування балів: якщо фіксація відвідуваності з технічних причин викликається двічі для одного і того ж запису, бал наховується лише один раз. При досягненні порогового значення студент може отримати безкоштовне заняття, і тест перевіряє, що відповідна кількість балів списується коректно, а запис на заняття фіксується без прив'язки до абонементу і не спричиняє нарахування боргу.

Результати виконання повного тестового набору підтверджують коректність реалізованих алгоритмів: усі 187 тестів пройшли успішно без жодної помилки. Модульні тести підтвердили правильність роботи окремих компонентів, інтеграційні - узгодженість взаємодії між моделями та базою

		Ступакова А.С.			ІПЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		69

даних, а HTTP-тести - коректність поведінки системи з точки зору реальних запитів користувача. Такий комплексний підхід до тестування забезпечує впевненість у надійності системи та значно знижує ризик появи прихованих помилок при подальшому розвитку програмного комплексу (Рис. 3.27).

```

✓ Create guest deduplicates by phone
✓ Create guest without phone always inserts
✓ Activate guest sets email and password
✓ Set active blocks and unblocks user
✓ Set trial used toggles flag
✓ Get all teachers active only excludes blocked

Visibility (Tests\Http\Visibility)
✓ Blocked teacher page returns 404 for guest
✓ Blocked teacher page returns 404 for student
✓ Blocked teacher page returns 404 for admin
✓ Active teacher page returns 200
✓ Student cabinet returns 200
✓ Teacher cabinet returns 200
✓ Admin cabinet redirects to admin
✓ Guest cabinet redirects to login
✓ Admin dashboard returns 200
✓ Student blocked from admin
✓ Teacher blocked from admin
✓ Guest admin redirects to login

OK (187 tests, 390 assertions)

```

Рис. 3.27 - Успішно пройдені тести

Висновки до третього розділу

У третьому розділі було описано інтерфейс, порядок встановлення та результати тестування вебплатформи танцювальної студії.

Розглянуто архітектуру розгортання системи, що складається з чотирьох вузлів: клієнтського браузера, вебсервера на базі Apache та PHP 8.2, сервера бази даних MySQL 8.0 та зовнішнього платіжного сервісу LiqPay. Описано покроковий порядок встановлення - від копіювання файлів і розгортання бази даних до налаштування конфігурації та прав доступу. Визначено три ролі користувачів системи (студент, викладач та адміністратор) з чітким розмежуванням доступу до функціональних розділів.

Описано інтерфейс користувача для кожної ролі. Публічна частина орієнтована на залучення нових клієнтів і надає загальну інформацію про студію та розклад занять. Особисті кабінети забезпечують студентам управління записами та абонементом, викладачам - перегляд розкладу та

відмітку відвідуваності, адміністратору - повний контроль над усіма процесами платформи.

Проведено автоматизоване тестування програмного комплексу за допомогою PHPUnit 11. Тести охоплюють контроль доступу, захист від CSRF-атак та SQL-ін'єкцій, фільтрацію завантажуваних файлів, бізнес-логіку записів на заняття, управління абонементом, нарахування боргу та роботу програми лояльності. Усі 187 тестів пройшли успішно, що підтверджує коректність реалізованих алгоритмів та надійність системи в цілому.

		Ступакова А.С.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		71

ВИСНОВКИ

Під час виконання кваліфікаційної роботи бакалавра було розроблено вебплатформу для танцювальної студії, яка автоматизує процеси що раніше велись вручну або через соціальні мережі, месенджери та електронні таблиці. Аналіз предметної області показав, що жоден із наявних на ринку продуктів не вирішує всі потреби студії комплексно: інформаційні сайти виконують лише презентаційну функцію, а універсальні сервіси запису не враховують специфіку навчального процесу.

На основі виявлених вимог спроектовано архітектуру системи за шаблоном MVC та розроблено базу даних із вісімнадцяти таблиць. Окремо пропрацьовано алгоритми найважливіших процесів - онлайн-оплати через LiqPay з верифікацією підпису callback-запиту, автоматичного списання занять з абонементу після фіксації відвідуваності, черги абонементів, розрахунку виплат викладачам та програми лояльності для студентів. Кожна з цих операцій супроводжується перевітками, що унеможливають некоректний стан даних.

Реалізований інтерфейс враховує потреби кожної ролі окремо. Студент записується на заняття, стежить за абонементом і балами лояльності. Викладач відмічає присутність і бачить свій баланс нарахувань. Адміністратор керує всім - від розкладу до виплат і модерації відгуків - з єдиної панелі.

Для перевірки коректності роботи системи написано 187 автоматизованих тестів, які охоплюють як безпеку — захист від підробки запитів, SQL-ін'єкцій, некоректних файлів — так і всю бізнес-логіку. Усі тести пройшли успішно, що підтверджує що система працює так як задумано.

		Ступакова А.С.			ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Чижмоторя О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		72

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Пасічник В. В. Організація баз даних та знань / В. В. Пасічник, В. А. Резніченко. – Київ : Видавнича група BHV, 2006. – 154 с
2. Пишемо Unit-тести на PHP: путівник PHPUnit та поради з досвіду. *dou*. URL: <https://dou.ua/lenta/articles/php-unit-testing/>.
3. Безпека веб-додатків: найкращі практики та вразливості. *Itproger*. URL: <https://itproger.com/ua/news/bezopasnost-veb-prilozheniy-luchshie-praktiki-i-uyazvimosti>.
4. CI/CD, DevOps та автоматизація у дипломному проєкті бакалавра. *Learn*. URL: https://learn.ztu.edu.ua/pluginfile.php/459399/mod_resource/content/3/diploma_supplement.html.
5. Діаграма розгортання. *Guru99*. URL: <https://www.guru99.com/uk/deployment-diagram-uml-example.html>.
6. *Dna Dance Studio*. URL: <https://www.dnadancestudio.lviv.ua/>.
7. *D.side Dance Studio*. URL: <https://ua.rivermall.dside.com.ua/>.
8. *EasyWeek*. URL: <https://easyweek.com.ua/>.
9. Застосування UML (частина 3). Діаграма класів - Class Diagram. *Duikt*. URL: https://duikt.edu.ua/ua/news-1-626-8002-zastosuvannya-uml-chastina-3-diagrama-klasiv----class-diagram_kafedra-kompyuternih-nauk-ta-informaciy-nih-tehnologiy.
10. Як створити ефективну програму лояльності для клієнтів. *Wedex*. URL: <https://wedex.com.ua/blog/yak-stvoryty-efektyvnu-programu-loyalnosti-dlya-kliiyentiv/>.
11. Ключі та нормалізація даних: основні нормальні форми (1nf, 2nf, 3nf, bcnf). *StudFiles*. URL: <https://studfile.net/preview/21632686/page:2/>.
12. *Liqpay*. URL: <https://www.liqpay.ua/doc>.
13. Побудова динамічних вебзастосунків за допомогою MVC. *HillelBlog*. URL: <https://blog.ithillel.ua/articles/building-dynamic-web-applications-using-mvc>.
14. Посібник з PHP. *Php*. URL: <https://www.php.net/manual/uk/>.

		Ступакова А.С.			ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		73

15. Розробка вебсайтів *Technologies MVC*. URL:
<https://brander.ua/technologies/mvc>.
16. Тестування PHP з PHPUnit. *FoxMinded*. URL:
<https://foxminded.ua/testuvannia-php-s-phpunit/>.
17. Що таке діаграма варіантів використання UML: символи, шаблони, інструмент і посібник. *Mindonmap*. URL:
<https://www.mindonmap.com/uk/blog/what-is-a-uml-use-case-diagram/>.
18. Advantages and Disadvantages of Java. *Geeksforgeeks*. URL:
<https://www.geeksforgeeks.org/java/advantages-and-disadvantages-of-java/>.
19. Comprehensive Guide to Git Pipelines. *Medium*. URL:
<https://medium.com/@programmingAi/comprehensive-guide-to-git-pipelines-0deb226b8c47>.
20. *Docker*. URL: <https://docs.docker.com>.
21. *Drawio*. URL: <https://www.drawio.com/>.
22. FOREIGN KEY Constraints. *Mysql*.. URL:
<https://dev.mysql.com/doc/refman/8.4/en/create-table-foreign-keys.html>.
23. Get started with GitLab CI/CD. *Gitlab*. URL: <https://docs.gitlab.com/ee/ci/>.
24. Documentation. *Mysql*. URL: <https://dev.mysql.com/doc/>.
25. password_hash. *Php*. URL: <https://www.php.net/manual/en/function.password-hash.php>.
26. PHPMailer – A full-featured email creation and transfer class for PHP. *Github*. URL: <https://github.com/PHPMailer/PHPMailer>.
27. Pros and Cons of Python Programming Language. *Serokell*. URL:
<https://serokell.io/blog/python-pros-and-cons>.
28. Singleton. *Refactoring guru*. URL: <https://refactoring.guru/design-patterns/singleton>.
29. Testing. *Liqpay*. URL: <https://www.liqpay.ua/en/doc/api/testing>.
30. Visual Studio Code documentation. *Visualstudio*. URL:
<https://code.visualstudio.com/docs>.

		Ступакова А.С.			ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		74

ДОДАТКИ

Діаграма варіантів використання системи

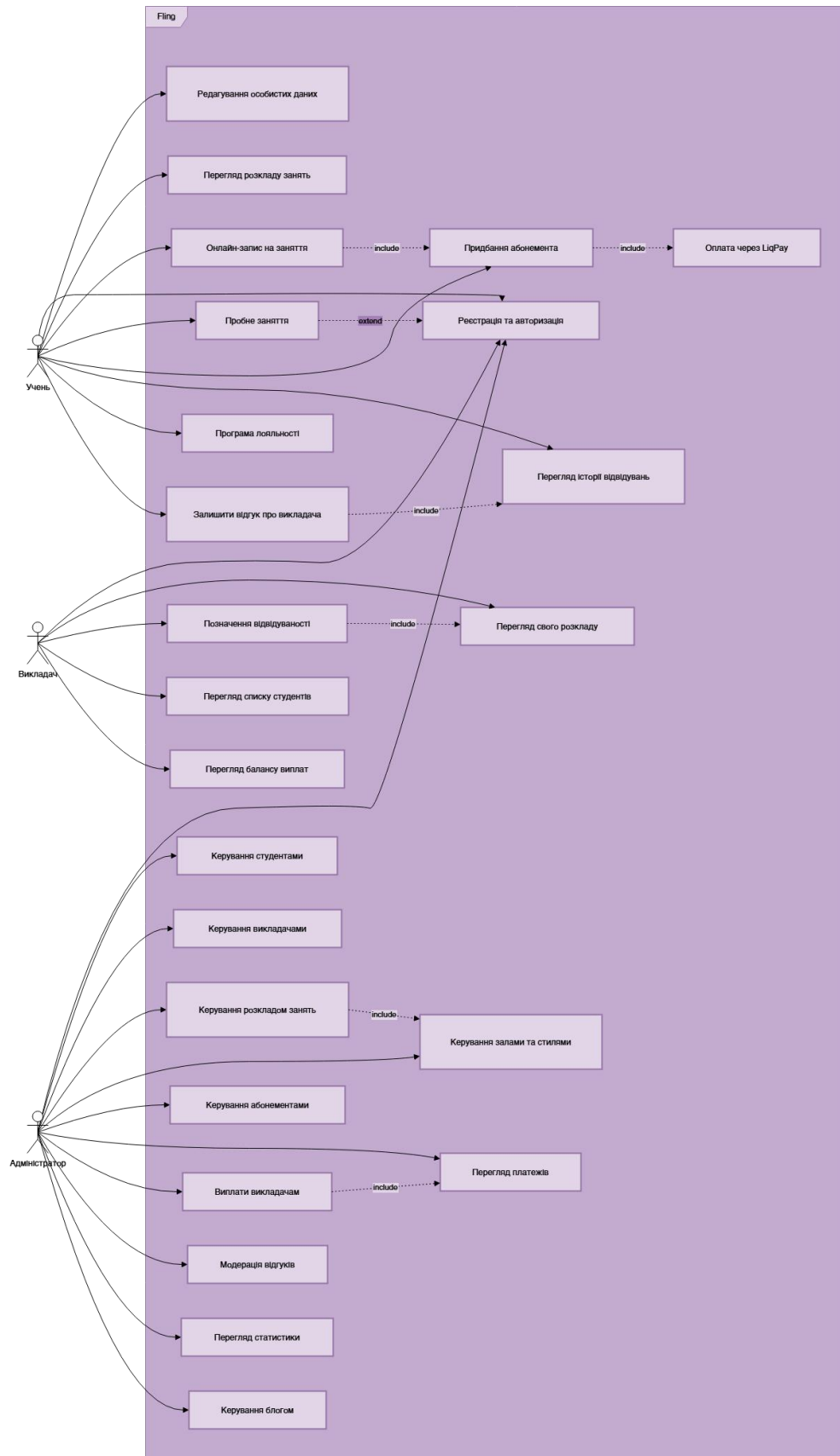


Рис. А.1. - Діаграма варіантів використання системи

Діаграма класів предметної області

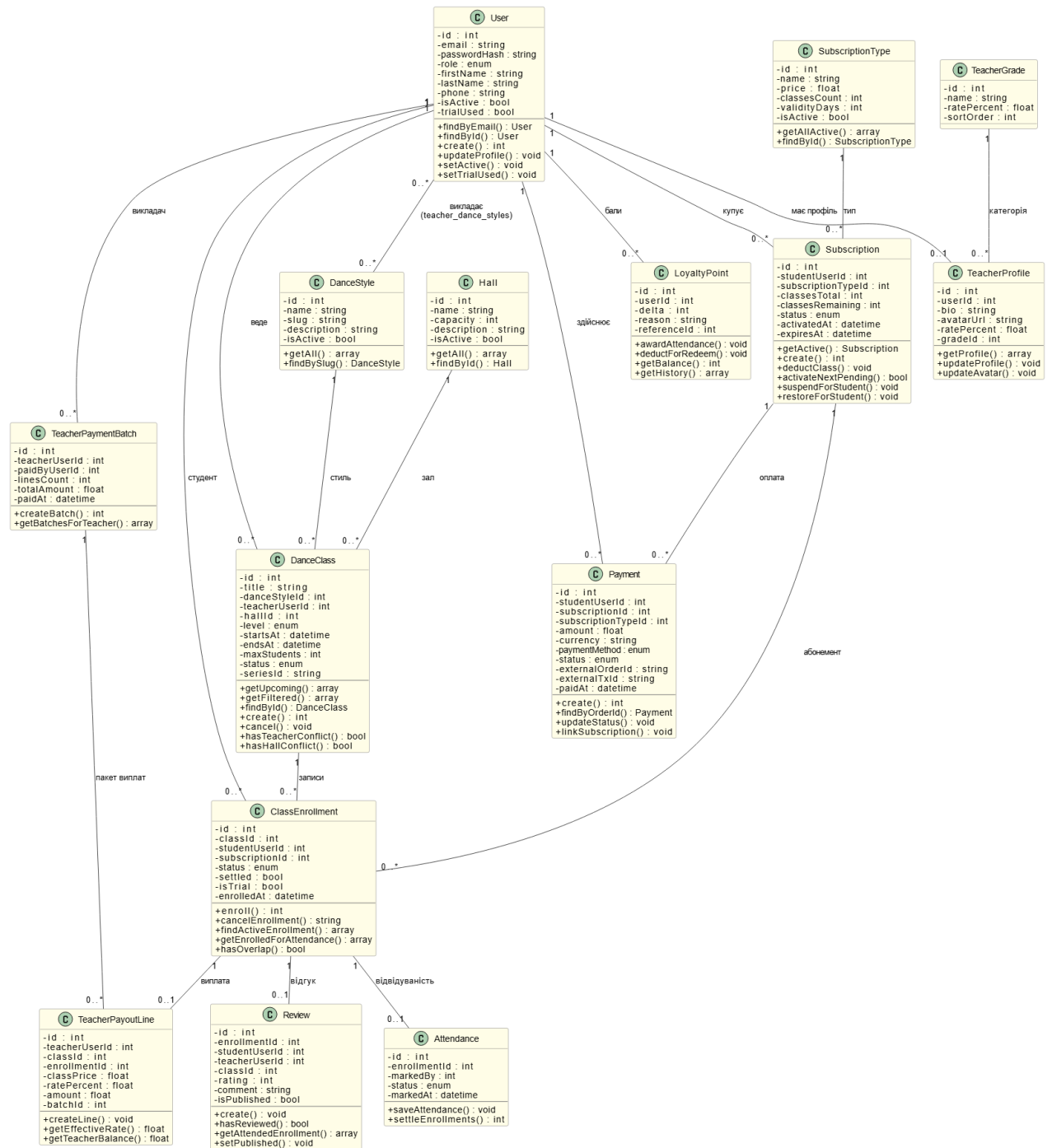


Рис. Б.1 - Діаграма класів предметної області

Архітектурна діаграма в межах MVC

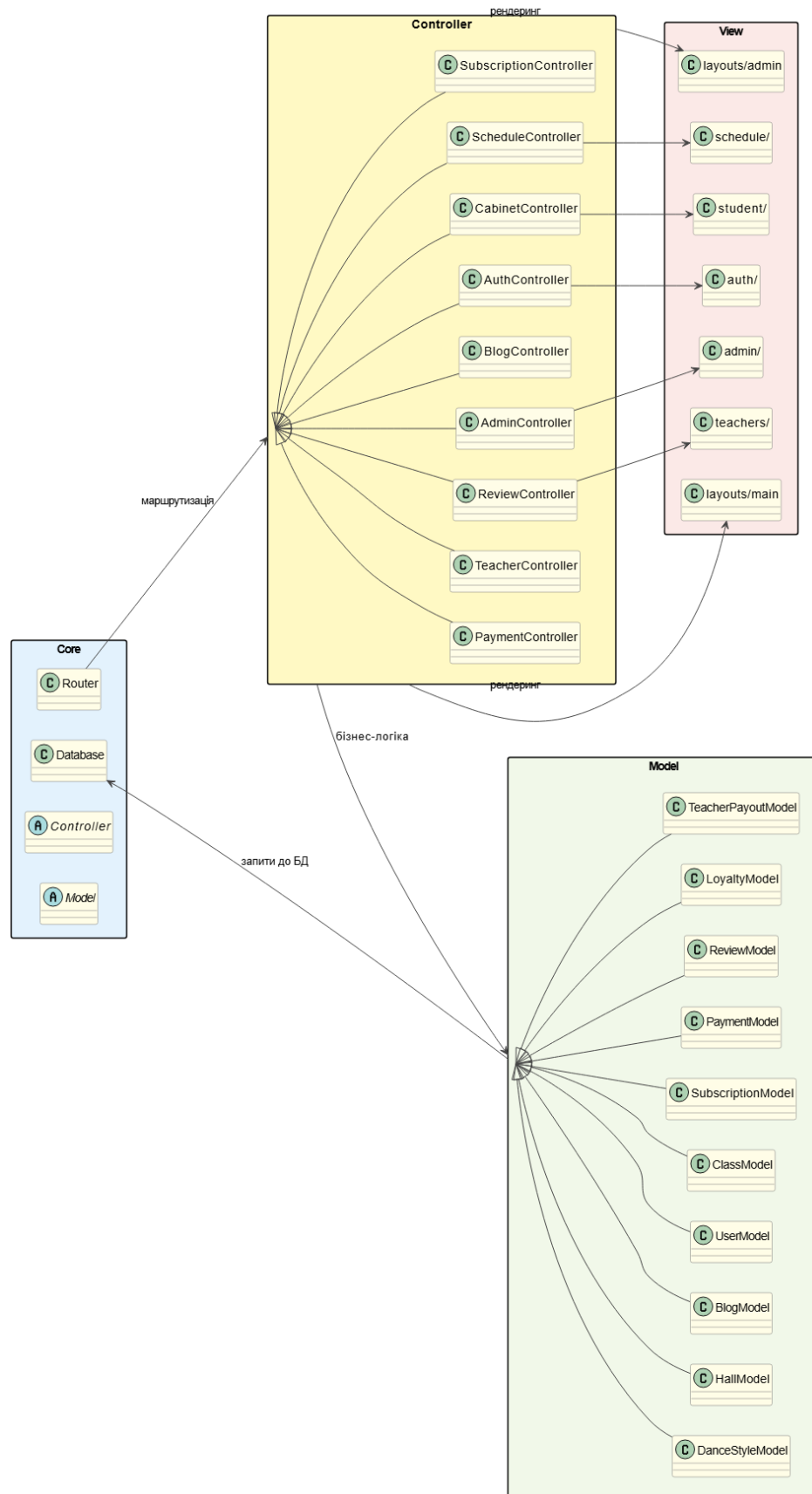


Рис. В.1 - Архітектурна діаграма в межах MVC

core/Router.php

```

<?php
namespace Core;

class Router
{
    private array $routes = [];

    private array $csrfExempt = [
        '/payment/callback',
    ];

    public function get(string $path, string $controller, string $action): void
    {
        $this->routes['GET'][$path] = ['controller' => $controller, 'action' =>
$action];
    }

    public function post(string $path, string $controller, string $action): void
    {
        $this->routes['POST'][$path] = ['controller' => $controller, 'action' =>
$action];
    }

    public function dispatch(string $uri, string $method): void
    {
        $uri      = strtok($uri, '?');
        $scriptDir = rtrim(str_replace('index.php', '', $_SERVER['SCRIPT_NAME']
?? '/'), '/');
        if ($scriptDir && $scriptDir !== '/' && str_starts_with($uri,
$scriptDir)) {
            $uri = substr($uri, strlen($scriptDir));
        }
        $uri = '/' . trim($uri, '/');
        if ($uri === '') $uri = '/';

        if ($method === 'POST' && !in_array($uri, $this->csrfExempt, true)) {

            $contentLength = (int)($_SERVER['CONTENT_LENGTH'] ?? 0);
            if ($contentLength > 0 && empty($_POST)) {
                $limit = ini_get('post_max_size');
            }
        }
    }
}

```

```

        throw new HttpException(413, '', 'Файл занадто великий. Максимальний розмір – ' . $limit . '. Стисніть зображення та спробуйте ще раз.');
```

```

    }

    $sessionToken = $_SESSION['csrf_token'] ?? '';
    $requestToken = $_POST['csrf_token'] ?? $_SERVER['HTTP_X_CSRF_TOKEN']
    ?? '';

    if (!$sessionToken || !hash_equals($sessionToken, $requestToken)) {
        throw new HttpException(419, '', 'Invalid CSRF token.');
```

```

    }

    }

    $routes = $this->routes[$method] ?? [];

    if (isset($routes[$uri])) {
        $this->run($routes[$uri], []);
        return;
    }

    foreach ($routes as $pattern => $handler) {
        if (!str_contains($pattern, '{')) continue;
        $regex = preg_replace('/\{[^\}]+\}/', '([^\/]*)', $pattern);
        $regex = '#^' . $regex . '$#u';
        if (preg_match($regex, $uri, $matches)) {
            array_shift($matches);
            $this->run($handler, $matches);
            return;
        }
    }
    $this->notFound();
}

private function run(array $handler, array $params): void
{
    $class = 'App\\Controllers\\' . $handler['controller'];
    $action = $handler['action'];
    $ctrl = new $class();
    $ctrl->$action(...$params);
}

private function notFound(): never
{
    throw new HttpException(404);
}
}

```

core/Controller.php

```

<?php
namespace Core;

```

```

class Controller
{
    protected function view(string $view, array $data = [], string $layout =
'main'): void
    {
        $viewPath = APP_ROOT . '/app/Views/' . str_replace('.', '/', $view) .
'.php';
        $data['csrfToken'] = $this->csrfToken();
        extract($data);

        ob_start();
        require $viewPath;
        $content = ob_get_clean();

        require APP_ROOT . '/app/Views/layouts/' . $layout . '.php';
    }

    protected function redirect(string $url): never
    {
        throw new HttpException(302, $url);
    }

    protected function json(mixed $data, int $code = 200): never
    {
        throw new HttpException($code, '', json_encode($data,
JSON_UNESCAPED_UNICODE), true);
    }

    protected function isPost(): bool
    {
        return $_SERVER['REQUEST_METHOD'] === 'POST';
    }

    protected function input(string $key, mixed $default = null): mixed
    {
        return $_POST[$key] ?? $_GET[$key] ?? $default;
    }

    protected function sanitize(string $val): string
    {
        return htmlspecialchars(trim($val), ENT_QUOTES, 'UTF-8');
    }

    protected function requireAuth(): void
    {
        if (!isset($_SESSION['user_id'])) {
            $this->redirect('/auth/login');
        }
    }
}

```

```

protected function requireRole(string ...$roles): void
{
    $this->requireAuth();
    if (!in_array($_SESSION['role'] ?? '', $roles)) {
        throw new HttpException(403, '', 'Доступ заборонено');
    }
}

protected function currentUser(): ?array
{
    return $_SESSION['user'] ?? null;
}

protected function csrfToken(): string
{
    if (empty($_SESSION['csrf_token'])) {
        $_SESSION['csrf_token'] = bin2hex(random_bytes(32));
    }
    return $_SESSION['csrf_token'];
}

protected function verifyCsrf(): void
{
    $token = $_POST['csrf_token'] ?? $_SERVER['HTTP_X_CSRF_TOKEN'] ?? '';
    if (!hash_equals($_SESSION['csrf_token'] ?? '', $token)) {
        http_response_code(419);
        exit('Invalid CSRF token.');
```

saveAttendance 3 ClassModel

```

public function saveAttendance(int $enrollmentId, string $status, int $markedBy):
void
{
    $this->ensureSettledColumn();
    $this->ensureIsTrialColumn();

    $this->execute(
        "INSERT INTO attendance (enrollment_id, status, marked_by, marked_at)
        VALUES (?, ?, ?, NOW())
        ON DUPLICATE KEY UPDATE status=VALUES(status),
marked_by=VALUES(marked_by), marked_at=NOW()",
        [$enrollmentId, $status, $markedBy]
    );

    $enrCheck = $this->queryOne(
        "SELECT settled, is_trial FROM class_enrollments WHERE id=?",
```

```

        [$enrollmentId]
    );
    if (!$enrCheck) return;
    if ($enrCheck['is_trial']) {
        $this->execute("UPDATE class_enrollments SET settled=1 WHERE id=? AND
settled=0", [$enrollmentId]);
        $this->accrueTeacherPayout($enrollmentId, null);
        return;
    }

    if ($status === 'excused') {
        $this->execute(
            "UPDATE class_enrollments SET settled=1 WHERE id=? AND
settled=0",
            [$enrollmentId]
        );
        return;
    }

    $enr = $this->queryOne(
        "SELECT settled, student_user_id, subscription_id FROM
class_enrollments WHERE id=? AND settled=0",
        [$enrollmentId]
    );
    if (!$enr) return;

    $studentId = (int)$enr['student_user_id'];
    $subId      = $enr['subscription_id'] ? (int)$enr['subscription_id'] :
null;

    if (!$subId) {
        $activeSub = $this->queryOne(
            "SELECT id FROM subscriptions
            WHERE student_user_id=? AND status='active' AND expires_at>? AND
classes_remaining>0
            ORDER BY created_at DESC LIMIT 1",
            [$studentId, date('Y-m-d H:i:s')]
        );
        $subId = $activeSub ? (int)$activeSub['id'] : null;
    }

    if ($subId) {
        $this->execute(
            "UPDATE subscriptions
            SET classes_remaining=GREATEST(CAST(classes_remaining AS
SIGNED)-1,0),
            status=IF(classes_remaining<=1,'exhausted',status)
            WHERE id=? AND classes_remaining < ?",
            [$subId, UNLIMITED_CLASSES]
        );
        $this->execute(

```

```

        "UPDATE class_enrollments SET settled=1, subscription_id=? WHERE
id=?",
        [$subId, $enrollmentId]
    );
} else {

    $this->execute(
        "UPDATE class_enrollments SET settled=1 WHERE id=?",
        [$enrollmentId]
    );
}

$this->accrueTeacherPayout($enrollmentId, $subId);
}

```

LiqPay

```

<?php
namespace App\Services;

class LiqPay
{
    private const CHECKOUT_URL = 'https://www.liqpay.ua/api/3/checkout';
    private const REQUEST_URL = 'https://www.liqpay.ua/api/request';

    private string $publicKey;
    private string $privateKey;

    public function __construct(string $publicKey, string $privateKey)
    {
        $this->publicKey = $publicKey;
        $this->privateKey = $privateKey;
    }

    public function buildData(array $params): string
    {
        $params['public_key'] = $this->publicKey;
        if (defined('LIQPAY_SANDBOX') && LIQPAY_SANDBOX) {
            $params['sandbox'] = '1';
        }
        return base64_encode(json_encode($params, JSON_UNESCAPED_UNICODE));
    }

    public function buildSignature(string $data): string
    {
        return base64_encode(sha1($this->privateKey . $data . $this->privateKey,
true));
    }

    public function verifySignature(string $data, string $signature): bool
    {

```

```

        return hash_equals($this->buildSignature($data), $signature);
    }

    public function decodeData(string $data): array
    {
        return json_decode(base64_decode($data), true) ?? [];
    }

    public function checkStatus(string $orderId): array
    {
        $data = $this->buildData([
            'version' => '3',
            'action'   => 'status',
            'order_id' => $orderId,
        ]);
        $signature = $this->buildSignature($data);

        $ch = curl_init(self::REQUEST_URL);
        curl_setopt_array($ch, [
            CURLOPT_POST           => true,
            CURLOPT_POSTFIELDS     => http_build_query(['data' => $data,
'signature' => $signature]),
            CURLOPT_RETURNTRANSFER => true,
            CURLOPT_TIMEOUT        => 15,
            CURLOPT_SSL_VERIFYPEER => !(defined('LIQPAY_SANDBOX') &&
LIQPAY_SANDBOX),
            CURLOPT_SSL_VERIFYHOST => (defined('LIQPAY_SANDBOX') &&
LIQPAY_SANDBOX) ? 0 : 2,
        ]);
        $response = curl_exec($ch);
        curl_close($ch);

        if (!$response) {
            return [];
        }
        return json_decode($response, true) ?? [];
    }

    public function getCheckoutUrl(): string
    {
        return self::CHECKOUT_URL;
    }
}

```

core/Model.php

```

<?php
namespace Core;

use PDO;

```

```

abstract class Model
{
    protected PDO $db;
    protected string $table = '';

    public function __construct()
    {
        $this->db = Database::getInstance();
    }

    protected function query(string $sql, array $params = []): array
    {
        $stmt = $this->db->prepare($sql);
        $stmt->execute($params);
        return $stmt->fetchAll();
    }

    protected function queryOne(string $sql, array $params = []): ?array
    {
        $stmt = $this->db->prepare($sql);
        $stmt->execute($params);
        $row = $stmt->fetch();
        return $row ?: null;
    }

    protected function execute(string $sql, array $params = []): int
    {
        $stmt = $this->db->prepare($sql);
        $stmt->execute($params);
        return (int) $this->db->lastInsertId();
    }

    protected function count(string $sql, array $params = []): int
    {
        $stmt = $this->db->prepare($sql);
        $stmt->execute($params);
        return (int) $stmt->fetchColumn();
    }
}

```