

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:

«Система адміністрування навчальної студії»

Виконала студентка групи ПЗ-21-4
КУРАЧ Олександра Адамівна

Керівник роботи:
Локтікова Тамара Миколаївна

Рецензент:
ВЛАСЕНКО Олег Васильович

Житомир – 2025

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

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії програмного
забезпечення

Тетяна ВАКАЛЮК

«17» лютого 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **КУРАЧ Олександра Адамівна**

Керівник роботи: **ЛОКТИКОВА Тамара Миколаївна**

Тема роботи: **«Система адміністрування навчальної студії»**,

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

Вихідні дані для роботи: **об'єктом дослідження є інформаційні системи адміністрування, що використовуються в освітній сфері, зокрема в приватних навчальних закладах та студіях, а предметом дослідження є програмні засоби та методи реалізації вебсистеми для управління навчальною студією, зокрема функціонал обліку клієнтів, розкладу занять, працівників і фінансових транзакцій.**

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Локтікова Т.М.	17.02.2025р.	17.02.2025р.
2	Локтікова Т.М.	17.02.2025р.	17.02.2025р.
3	Локтікова Т.М.	17.02.2025р.	17.02.2025р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з системи адміністрування навчальної студії та пояснювальної записки. Пояснювальна записка до випускної роботи містить 77 сторінок, 59 ілюстрацій та 11 таблиць.

Виконано розробку системи адміністрування навчальної студії з використанням сучасних вебтехнологій.

Архітектура системи реалізована за принципом клієнт-серверної взаємодії: серверна частина побудована з використанням фреймворку Laravel, клієнтська — на базі React. Взаємодія між модулями здійснюється через REST API. У проєкті також використовується Docker для контейнеризації та Telegram-бот для сповіщення клієнтів.

Вебінтерфейс забезпечує зручну роботу для адміністраторів студії, дозволяючи створювати розклад, керувати фінансами, відстежувати статистику та проводити аналітику ефективності.

КЛЮЧОВІ СЛОВА: АДМІНІСТРУВАННЯ, НАВЧАЛЬНА СТУДІЯ, РОЗКЛАД ЗАНЯТЬ, ФІНАНСИ, DOCKER, LARAVEL, REACT, TELEGRAM-БОТ.

					ІПЗ.КР.Б-121-25-ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Система адміністрування навчальної студії. Пояснювальна записка	Літ.	Арк.	Аркушів
Розроб.		Курач О.А.						
Керівник.		Локтікова Т.М.					3	77
Н. контр.		Тичина О.П.				ФІКТ Гр. ІПЗ-21-4		
Зав. каф.		Вакалюк Т.А.						

ABSTRACT

Bachelor's Qualification Thesis consists of an educational studio administration system and an explanatory report. The explanatory report contains 77 pages, 59 illustrations, and 11 tables. The goal of the work is to develop a functional system for managing an educational studio using modern web development technologies.

The system architecture follows the client-server interaction model: the server side is built using the Laravel framework, while the client side is based on React. Module interaction is implemented through a REST API. The project also uses Docker for containerization and a Telegram bot for client notifications.

The web interface provides a convenient working environment for studio administrators, enabling them to create schedules, manage finances, track statistics, and perform efficiency analysis.

KEYWORDS: ADMINISTRATION, DOCKER, FINANCES, LARAVEL, LEARNING STUDIO, REACT, SCHEDULE OF LESSONS, TELEGRAM BOT.

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		4

ЗМІСТ

РЕФЕРАТ.....	3
ABSTRACT.....	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	6
ВСТУП.....	7
РОЗДІЛ 1 АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ СИСТЕМИ АДМІНІСТРУВАННЯ НАВЧАЛЬНОЇ СТУДІЇ.....	9
1.1 Постановка задачі	9
1.2 Аналіз аналогів програмного продукту	10
1.3 Вибір архітектури системи адміністрування навчальної студії	16
1.4 Обґрунтування вибору інструментальних засобів	18
Висновки до першого розділу	19
РОЗДІЛ 2 ПРОЄКТУВАННЯ СИСТЕМИ АДМІНІСТРУВАННЯ	21
2.1 Визначення варіантів використання та об'єктно-орієнтованої структури системи	21
2.2 Розробка бази даних системи	24
2.3 Проєктування та реалізація алгоритмів роботи системи	33
2.4 Реалізація функціоналу системи	38
Висновки до другого розділу	43
РОЗДІЛ 3 ІНТЕРФЕЙС І ПОРЯДОК РОБОТИ З СИСТЕМОЮ АДМІНІСТРУВАННЯ НАВЧАЛЬНОЇ СТУДІЇ.....	45
3.1 Порядок встановлення та налаштування параметрів системи	45
3.2 Структура інтерфейсу.....	46
3.3. Тестування роботи програмного комплексу	68
Висновки до третього розділу	73
ВИСНОВКИ	74
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	75
ДОДАТКИ	77

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – прикладний програмний інтерфейс

БД – база даних

ПЗ – програмне забезпечення

СУБД – система управління базами даних

SDK – набір для розробки програмного забезпечення (software development kit)

MVC – схема розділення даних додатків і керуючої логіки на три окремі компоненти Model-View-Controller

CRUD – основні операції, що використовуються для управління даними (Create, Read, Update, Delete)

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		6

ВСТУП

Актуальність теми пов'язана з тим, що у сучасному суспільстві інформаційні технології дедалі більше інтегруються у всі сфери життя, зокрема й в освітній процес. Навчальні студії, які надають послуги з додаткової освіти (танці, музика, спорт, іноземні мови, тощо), потребують ефективних інструментів для організації своєї діяльності. Зростаюча кількість клієнтів, різноманіття курсів, гнучкий розклад, взаємодія з викладачами та фінансові операції — все це вимагає централізованого та зручного способу адміністрування. Багато студій продовжують вести облік клієнтів, викладачів, занять та оплат вручну або за допомогою окремих, неінтегрованих сервісів (електронні таблиці, месенджери, мобільні застосунки для розкладу тощо). Такий підхід призводить до втрати інформації, дублювання даних, помилок в обліку та ускладнення управління. У зв'язку з цим постає потреба у створенні єдиної системи адміністрування, яка б об'єднувала всі ключові функції: планування розкладу, облік працівників і клієнтів, фінансові операції, а також зберігала зручність у використанні. Саме тому розробка вебзастосунку для адміністрування навчальної студії є актуальним завданням, яке спрямоване на вирішення нагальних потреб малого та середнього освітнього бізнесу, забезпечення ефективного управління ресурсами та підвищення якості обслуговування клієнтів.

Метою випускної кваліфікаційної роботи бакалавра є розробка системи адміністрування навчальної студії з використанням сучасних вебтехнологій, яка забезпечить зручність у використанні, ефективність управління, а також можливість масштабування та адаптації до потреб різних типів навчальних закладів. Автоматизація основних адміністративних процесів сприяє зменшенню навантаження на персонал, оптимізації часу виконання рутинних завдань, зниженню кількості помилок, а також покращенню взаємодії між адміністрацією, працівниками та клієнтами студії. Запровадження такої системи дозволяє не лише підвищити загальну ефективність управління, але й створює нові можливості для розвитку бізнесу, аналізу діяльності та масштабування освітніх послуг. Запропонована система має бути універсальною, з гнучкою структурою, яка

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		7

дозволяє легко адаптувати її під специфіку різних типів організацій — навчальних студій, приватних шкіл, курсів тощо.

Для досягнення цієї мети потрібно виконання таких **завдань**: огляд й аналіз аналогів програмного продукту, вибір архітектури системи, вибір й обґрунтування інструментальних засобів реалізації системи, побудова та проєктування системи, створення зручного, функціонального та адаптивного інтерфейсу, що дозволяє адміністраторам студії легко управляти даними, здійснювати моніторинг навчального процесу, оперативно вносити зміни до розкладу, обліковувати платежі та формувати звітність.

Об’єктом дослідження є інформаційні системи адміністрування, що використовуються в освітній сфері, зокрема в приватних навчальних закладах та студіях.

Предметом дослідження є програмні засоби та методи реалізації вебсистеми для управління навчальною студією, зокрема функціонал обліку клієнтів, розкладу занять, працівників і фінансових транзакцій.

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		8

РОЗДІЛ 1 АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ СИСТЕМИ АДМІНІСТРУВАННЯ НАВЧАЛЬНОЇ СТУДІЇ

1.1 Постановка задачі

Кваліфікаційна робота бакалавра спрямована на створення системи адміністрування навчальної студії з використанням сучасних технологій, що забезпечують зручність у користуванні та можливість подальшого масштабування.

Автоматизація процесів полегшує навантаження на людей і бізнес та дозволяє відкрити нові можливості для обох сторін. Створення такої платформи має полегшити комунікацію та зменшити навантаження на адміністрацію.

Створення універсальної системи адміністрування з можливістю легкої адаптації під різні студії, школи чи системи курсів має вирішити проблеми, описані у табл. 1.1

Таблиця 1.1

Проблема	Кого зачіпає	Запропоноване рішення
Необхідність особистого контакту з адміністратором для запису на заняття	Відвідувачі, Адміністратор	Онлайн-система для самостійного запису на заняття
Ручне надсилання нагадувань про заняття або оплати	Адміністратор	Впровадження автоматичної системи сповіщень
Розпорошеність інформації між різними програмами, таблицями та документами	Адміністратор	Єдина система для зберігання та управління усією інформацією
Необхідність ручного ведення документообігу	Адміністратор	Автоматизована генерація звітів і можливість аналітики
Обмеженість способів оплати занять	Відвідувачі	Додавання функції онлайн-оплати через зручні платіжні сервіси

Для реалізації зазначеної мети передбачено проходження низки етапів, що охоплюють увесь цикл розробки програмного забезпечення:

1. Аналіз предметної області, виявлення проблем та визначення потреб користувачів.
2. Проектування бази даних та структури майбутньої системи.
3. Налаштування середовища розробки з використанням сучасних інструментів.
4. Розробка архітектури та серверної частини застосунку.
5. Створення дизайну інтерфейсу користувача (frontend).
6. Реалізація клієнтської частини з урахуванням інтерактивних функцій.

Тестування функціоналу системи та його аналіз здійснюються паралельно з розробкою для забезпечення своєчасного виявлення помилок та їх усунення.

У результаті реалізації проєкту буде створено повноцінну інформаційну систему, що надає користувачам можливість реєстрації на заняття, здійснення оплати, перегляду особистої інформації та отримання автоматичних сповіщень. Для адміністрації студії система забезпечує інструменти для управління розкладом, клієнтами, фінансовими операціями та звітністю, що значно полегшує процес адміністрування.

1.2 Аналіз аналогів програмного продукту

Для розробки ефективної системи адміністрування навчальної студії важливо проаналізувати існуючі CRM-рішення, які вже використовуються в суміжних галузях — йога-студіях, танцювальних школах, освітніх платформах тощо. Аналіз таких рішень дозволяє зрозуміти потреби користувачів і визначити ключові компоненти майбутнього продукту.

OtmetchalkaCRM [14] – є найпопулярнішою українською платформою. Серед основних функцій системи варто виділити автоматизований запис на заняття — як групові, так і індивідуальні. Це значно спрощує процес реєстрації клієнтів і зменшує навантаження на адміністраторів. Передбачено створення особистих кабінетів для кожного клієнта. У них користувачі можуть переглядати розклад,

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		

власну історію відвідувань, оплату та інші персоналізовані дані. Система також підтримує автоматичні нагадування про заняття, повідомлення про зміни в розкладі та вітання з нагоди свят, що сприяє покращенню комунікації з клієнтами та підвищенню їхньої залученості. Реалізовано облік продажів товарів та складський облік, що дозволяє вести повний контроль за матеріальними ресурсами студії. Для адміністраторів і менеджерів доступна потужна система аналітики — звіти щодо відвідуваності, фінансових показників, популярності курсів тощо. Крім того, система може містити відеоінструкції для користувачів, які допомагають швидко ознайомитися з функціоналом та розпочати роботу.

На рис. 1.1 представлено головну сторінку системи OtmechalkaCRM.

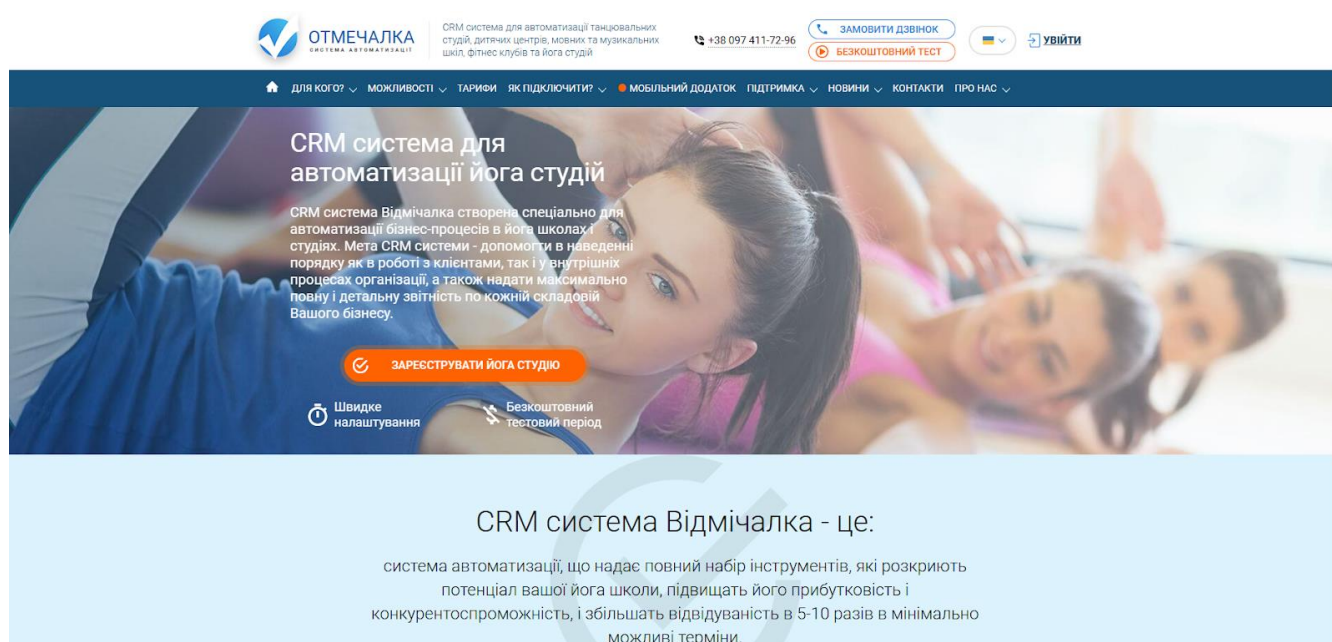


Рисунок 1.1 –Головна сторінка OtmechalkaCRM

OneBox [13] – це хмарна платформа, серед ключових функцій якої — збір і зберігання даних клієнтів, що дозволяє створювати повноцінні клієнтські профілі з історією занять, оплат та взаємодії. Це дає змогу краще розуміти потреби кожного користувача та будувати персоналізовану взаємодію. Важливою складовою є планування розкладу занять із можливістю відстеження відвідуваності. Це допомагає адміністраторам ефективно управляти навантаженням викладачів, заповненістю груп і своєчасно вносити зміни. Система підтримує автоматизацію прийому платежів та контроль їх виконання. Це мінімізує ручну роботу, забезпечує прозорість фінансових операцій та своєчасну звітність. Передбачено можливість

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
						11
Змн.	Арк.	№ докум.	Підпис	Дата		

масових або індивідуальних розсилок повідомлень — як нагадувань, так і інформації про зміни в розкладі. Це дозволяє тримати зв'язок з клієнтами та швидко реагувати на зміни. Функціонал обліку уроків полегшує організацію навчального процесу та сприяє покращенню комунікації між викладачами й учнями. Нарешті, завдяки статистиці й аналітиці можна відстежувати динаміку відвідуваності, обсяг оплат та ефективність окремих курсів. На рис. 1.2 зображено сторінку OneBox для йога студій.

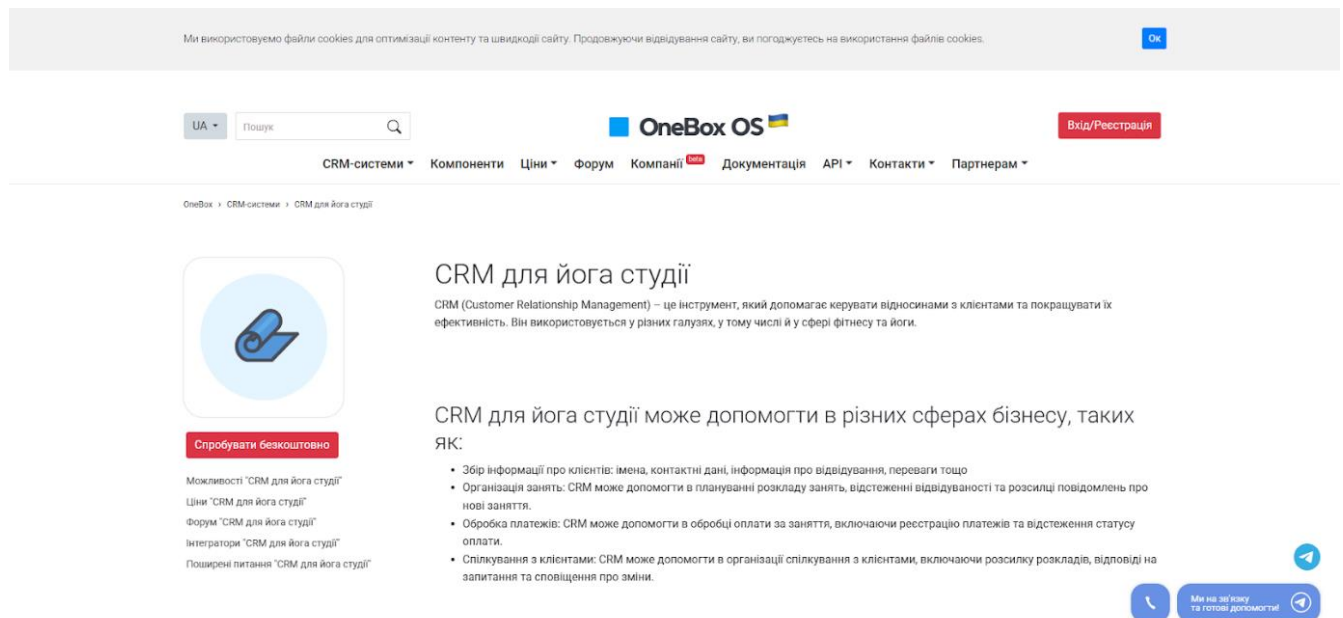


Рисунок 1.2 – Сторінка OneBox для йога студій

CRM Dance [5] - це сучасна хмарна CRM-система, що надає інструменти для повноцінного управління розкладом занять, фінансовими операціями та аналітикою. Це дозволяє адміністраторам і менеджерам контролювати всі ключові процеси студії в одному місці. Передбачено облік відвідуваності та можливість ефективної взаємодії зі студентами. Це допомагає підтримувати дисципліну, бачити загальну картину участі та реагувати на проблеми своєчасно. Інтерфейс розроблений таким чином, щоб забезпечити максимально просту роботу з клієнтами. Усе, що стосується запису, оплати, змін у розкладі чи комунікації, реалізовано інтуїтивно і швидко. Клієнти мають зручний доступ до свого розкладу та динаміки навчального прогресу, що підвищує їхню залученість і мотивацію. Особлива увага приділена зручності роботи для викладача — уся необхідна інформація доступна у кілька кліків, що дозволяє зосередитися на навчальному

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
						12
Змн.	Арк.	№ докум.	Підпис	Дата		

процесі, а не на адміністративних дрібницях. На рис. 1.3 зображено головну сторінку CRM Dance.

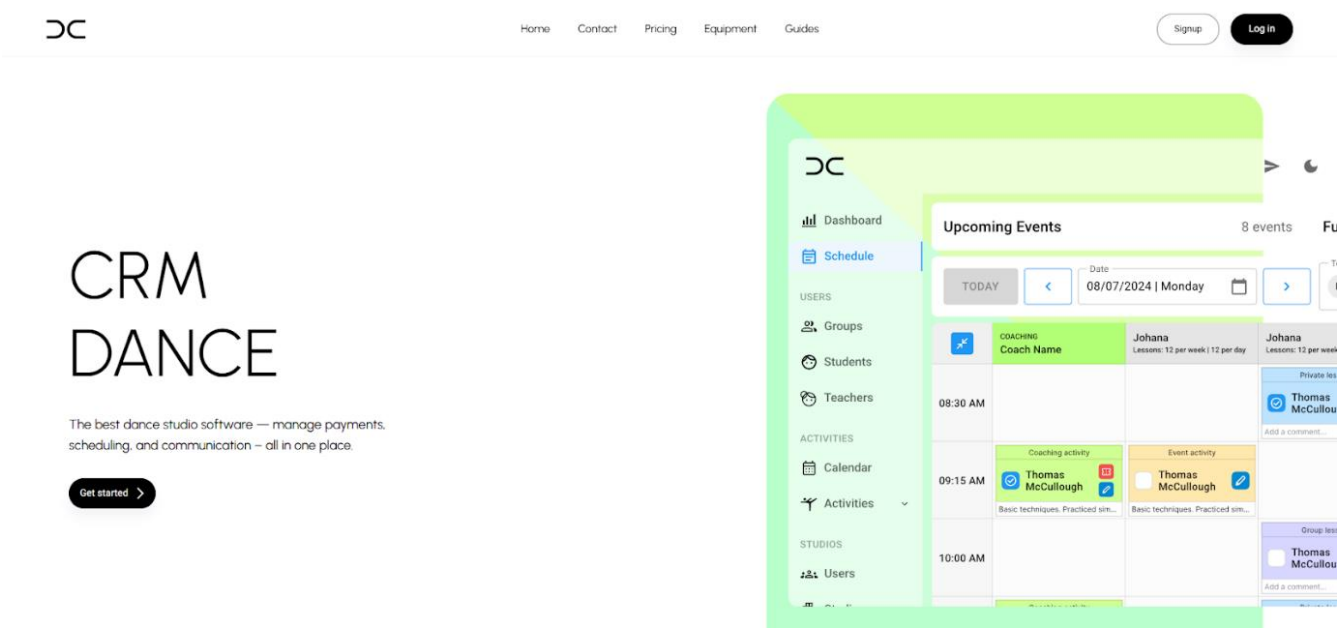


Рисунок 1.3 – Головна сторінка CRM Dance

EasyBusy [7] - це українська хмарна CRM-система, що дозволяє ефективно керувати продажами, виконанням завдань та обробкою клієнтських баз. Це забезпечує чіткий контроль за бізнес-процесами та комунікацією з клієнтами. Передбачена гнучка інтеграція з популярними платформами — Instagram та Telegram. Це дає змогу використовувати чат-боти, сповіщення та живі чати для швидкої взаємодії з аудиторією. Підтримуються контактні форми та API-інтеграції з сайтом студії, що дає можливість автоматизувати збір заявок, звернень та зворотного зв'язку. Додатково реалізовано інтеграцію з Google Forms та UniTalk — для збору анкет, оцінок або запису на консультації. Функція автоматичного запису дзвінків через Vinotel дозволяє зберігати повну історію спілкування з клієнтами та підвищує якість обслуговування. Клієнти можуть самостійно бронювати заняття через сайт, що зручно і знижує навантаження на адміністрацію. Також система підтримує збереження юридичної інформації клієнтів із реєстрів — це корисно для ведення договірної документації або співпраці з компаніями. На рис. 1.4 представлено головну сторінку сайту EasyBusy.

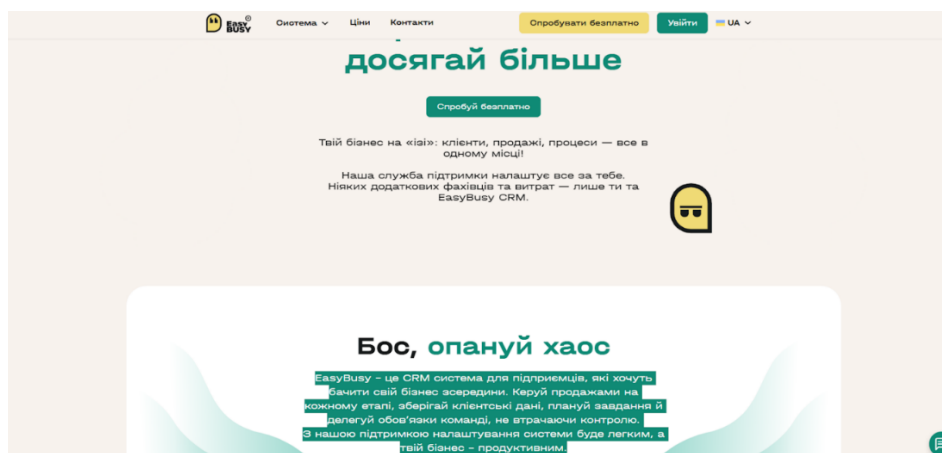


Рисунок 1.4 – Головна сторінка сайту EasyBus

NeoDove [11] - це система, спеціалізована на хмарному зберіганні клієнтських даних із можливістю швидкого доступу з будь-якого пристрою. Це гарантує безпеку, зручність і актуальність інформації. Передбачено зручне управління лідами, командою та розкладом, що дозволяє адміністраторам і менеджерам ефективно координувати всі процеси студії. Автоматизація маркетингу та масових розсилок допомагає підтримувати зв'язок із клієнтами, повідомляти їх про новини, акції та зміни у розкладі без зайвих зусиль. Функціонал аналітики і звітності дає змогу контролювати ключові показники — відвідуваність, фінанси, успішність кампаній — і приймати зважені управлінські рішення. Централізація всієї інформації про клієнтів у єдиній системі забезпечує зручний доступ до історії записів, оплат та взаємодії. У результаті — значна оптимізація робочих процесів та економія часу як для адміністративного персоналу, так і для викладачів. На рис. 1.5 представлена сторінка NeoDove для танцювальних студій.

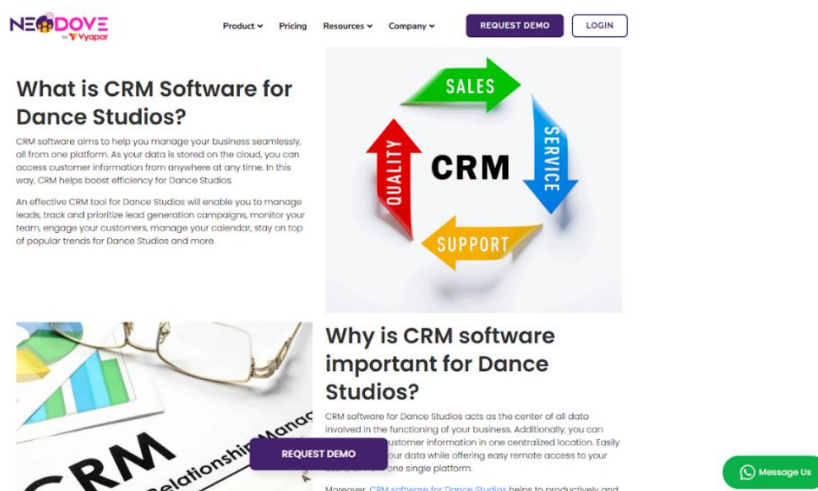


Рисунок 1.5 – Сторінка NeoDove для танцювальних студій

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		14

Аналіз сучасних CRM-систем для студій показує чіткі загальні тенденції в автоматизації цього бізнесу. Майже всі рішення зосереджені на декількох ключових функціях: онлайн-бронювання занять, облік відвідуваності, ведення клієнтських баз, автоматизація платежів та розсилок, інтеграція з месенджерами (Telegram, Instagram), а також формування звітності й аналітики для керування. Простота у використанні для адміністратора, викладача та клієнта — ще один важливий тренд, тому багато систем мають особисті кабінети або мобільні версії. Деякі CRM, як-от EasyBusy, також роблять акцент на глибокій інтеграції з зовнішніми сервісами (Google Forms, API, Binotel) та підтримці автоматичних чат-ботів, що особливо актуально для малого бізнесу з обмеженим ресурсом на персонал. Структуроване порівняння описане у табл. 1.2.

Таблиця 1.2

CRM-система	Otmechalka	1b.app	CRM.Dance	EasyBusy	Neodove
Розклад занять	+	+	+	+	+
Запис клієнтів	+	+	+	+	+
Особистий кабінет	+	-	+	-	-
Платежі та фінанси	+	+	+	+	+
Автоматичні нагадування	+	+	+	+	+
Аналітика та звітність	+	+	+	+	+
Інтеграції з Telegram, Instagram, тощо	-	-	-	+	+
Контроль працівників	-	-	+	-	+

У власній розробці доцільно реалізувати низку ключових функцій. По-перше, система повинна забезпечувати реєстрацію та авторизацію користувачів із надійною автентифікацією. Реєстрація має бути швидкою, але ефективною, надаючи доступ до особистого кабінету з індивідуальними налаштуваннями. Важливою складовою є управління розкладом занять: створення, редагування та видалення занять, призначення інструкторів та студій, відображення розкладу в особистому кабінеті користувача. Наступною ключовою функцією є запис на заняття, який повинен бути автоматизованим, дозволяти відвідувачам самостійно

реєструватися на групові або індивідуальні заняття, враховувати обмеження за кількістю місць та автоматично оновлювати статус груп.

Також важливим є модуль обліку платежів та фінансової аналітики. Він має включати відстеження платежів, облік заборгованостей, автоматичний розрахунок заробітної плати, формування звітності, надсилання нагадувань про оплату та можливість здійснення онлайн-платежів через Stripe або LiqPay. Окрім цього, система повинна забезпечувати надсилання повідомлень та нагадувань, зокрема автоматичні сповіщення про заняття, зміни в розкладі, скасування, підтвердження записів та повідомлення про оплату.

Адміністрування інструкторів є окремим напрямком і включає управління їхніми даними, призначення курсів, контроль навантаження та перегляд історії занять. Також доцільним є впровадження модуля модерації та управління контентом, який дозволить переглядати та редагувати профілі користувачів, модерувати відгуки та скарги, здійснювати пошук і фільтрацію курсів та занять за різними критеріями: інструктор, дата, рівень складності, ключові слова.

Для підтримки ефективного документообігу система повинна мати аналітичний модуль, який збиратиме статистику відвідуваності, завантаженості студії, популярності напрямків і курсів, формуватиме фінансові звіти та оцінюватиме ефективність інструкторів.

У результаті повинна бути створена комплексна система з окремих взаємопов'язаних модулів, яка забезпечить повноцінну підтримку функціонування студії, буде доступною для всіх працівників та зручною у користуванні для відвідувачів.

1.3 Вибір архітектури системи адміністрування навчальної студії

Архітектура системи адміністрування студії була обрана з урахуванням потреби забезпечити стабільну, гнучку, зручну та масштабовану платформу, яка дозволяє ефективно управляти заняттями, розкладом, працівниками, клієнтами, фінансами та внутрішніми процесами навчальної студії [19]. Основу системи становить клієнт-серверна архітектура з поділом на фронтенд (інтерфейс

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		16

користувача), бекенд (серверна логіка) та базу даних. Така структура дає змогу розмежувати обов'язки між компонентами, прискорити розробку та забезпечити багаторазове використання API для доступу до різних каналів — веб-інтерфейсу, мобільного застосунку, чи сторонніх інтеграцій.

Фронтенд реалізується на React.js із використанням сучасних бібліотек для побудови адаптивного та інтерактивного інтерфейсу (зокрема drag-and-drop розклад занять, модальні вікна, фільтрація, тощо) [21]. Бекенд створено за допомогою Laravel — популярного PHP-фреймворку, який дозволяє швидко реалізовувати RESTful API, обробку форм, а також керування користувачами, ролями, правами доступу, транзакціями та аналітикою [25]. Для зберігання даних використовується реляційна база даних MySQL, в якій фіксуються всі ключові сутності системи: користувачі, працівники, уроки, розклад, платежі, групи, фінансові транзакції тощо.

В рамках проєкту реалізовано повноцінну адміністративну панель, що дозволяє керувати співробітниками (інструкторами та адміністраторами), клієнтами, записом на заняття, платежами, групами, курсами та студіями. Особистий кабінет користувача дає змогу бачити власний розклад, історію відвідувань, оплат, повідомлень тощо. Модулі формуються за принципом логічної відокремленості: управління розкладом, фінансова аналітика, адміністрування інструкторів, повідомлення та модерація контенту. Таке розділення дозволяє масштабувати систему — при потребі можна винести модуль, наприклад, фінансової звітності чи запису на заняття в окремий сервіс.

На першому етапі розробки було обрано підхід модульного моноліту — усі компоненти існують в межах одного застосунку, проте чітко структуровані за функціональністю. Це забезпечує швидку реалізацію MVP, а в майбутньому — можливість переведення певних частин у мікросервіси. API реалізовано у вигляді REST-запитів [20] із захистом через Laravel Sanctum або JWT-токени, а також реалізована рольова система доступу (адміністратор, інструктор, клієнт), що дозволяє показувати кожному користувачу лише відповідні інтерфейси й дані.

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
						17
Змн.	Арк.	№ докум.	Підпис	Дата		

Для забезпечення безпеки використовуються такі інструменти: HTTPS, контроль сесій, захист форм через CSRF, обмеження доступу до критичних дій, журналювання змін адміністраторами. Планується розгортання на хмарному сервері з використанням Docker-контейнерів для фронтенду, бекенду та бази даних. Система автоматичного розгортання (CI/CD) забезпечить швидке оновлення, а регулярне резервне копіювання бази даних дозволить уникнути втрати інформації [18].

Завдяки обраній архітектурі система є достатньо гнучкою для впровадження додаткового функціоналу: нотифікацій, підключення Telegram-бота або мобільного застосунку, інструментів для онлайн-оплати (Stripe, LiqPay [9]), автоматичної генерації зарплат, аналітики ефективності інструкторів, тощо. Такий підхід дозволяє будувати сучасну, прозору та зручну систему керування навчальною студією.

1.4 Обґрунтування вибору інструментальних засобів

Для реалізації системи адміністрування навчальної студії було обрано сучасний стек технологій, який забезпечує надійність, масштабованість та зручність у розробці. Серверна частина проєкту реалізована за допомогою PHP-фреймворку Laravel, який є одним із найпопулярніших рішень у галузі веброзробки [17]. Laravel забезпечує швидкий старт завдяки вбудованим механізмам для аутентифікації, авторизації, роботи з базою даних через ORM (Eloquent), маршрутизації, валідації форм, керування подіями та обробки черг. Його модульна структура дозволяє легко додавати нові функціональні блоки та інтегрувати сторонні сервіси — наприклад, Stripe для оплати або Telegram API [16] для надсилання повідомлень. Широка екосистема пакетів та офіційна документація роблять Laravel оптимальним вибором для побудови повноцінного адміністративного інструменту студії.

Для зберігання інформації про користувачів, заняття, групи, платежі, розклад та інші сутності обрана реляційна система управління базами даних MySQL. Вона є перевіреним рішенням, сумісним із Laravel та підтримує складні SQL-запити,

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
						18
Змн.	Арк.	№ докум.	Підпис	Дата		

транзакції, індекси, зв'язки між таблицями, а також має добру продуктивність як для читання, так і для запису даних. MySQL зручно налаштовується в локальному середовищі розробника або на хмарному сервері, підтримує резервне копіювання, масштабування та реплікацію, що робить її гнучким і надійним компонентом системи [28].

Клієнтська частина створена за допомогою JavaScript-бібліотеки React, яка дозволяє будувати швидкі, адаптивні та масштабовані інтерфейси. React працює за принципом компонентного підходу — це дає змогу створювати повторно використовувані UI-елементи, які легко підтримувати й масштабувати. Для проєкту, де є складна взаємодія з розкладом занять, таблицями транзакцій, формами для створення та редагування даних, а також численними фільтрами, React є ідеальним рішенням. Крім того, бібліотека має широку підтримку серед розробників, велику кількість готових компонентів і рішень (наприклад, Material UI для інтерфейсу, React Query для роботи з API, Chart.js для графіків), що пришвидшує розробку та зменшує ризики.

Загалом, поєднання Laravel, MySQL та React дозволяє побудувати надійну, модульну та зручну для користувача вебсистему, яка охоплює всі потреби навчальної студії: від керування розкладом та записом на заняття до фінансового обліку, аналітики та комунікації з клієнтами. Такий технологічний стек забезпечує гнучкість у подальшому розвитку системи та адаптацію до нових функціональних потреб.

Висновки до першого розділу

У першому розділі було проведено комплексний аналіз предметної області, визначено ключові проблеми та сформульовано основні цілі для створення системи адміністрування навчальної студії. Було виявлено основні потреби користувачів і працівників студії, зокрема — необхідність автоматизації процесів управління розкладом, запису на заняття, обліку платежів, ведення фінансової аналітики, а також ефективної комунікації через повідомлення та нагадування. Проведене дослідження існуючих аналогів дозволило виявити їхні сильні та слабкі сторони,

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
						19
Змн.	Арк.	№ докум.	Підпис	Дата		

що стало основою для формування чітких функціональних вимог до майбутньої системи.

На основі зібраної інформації було обґрунтовано вибір архітектурного підходу — модульного моноліту з можливістю подальшого масштабування, що дозволяє швидко створити мінімально життєздатний продукт (MVP) [29] і надалі розвивати його без необхідності повного перепроєктування. Також було обґрунтовано вибір інструментальних засобів, що найкраще відповідають вимогам проєкту: Laravel для бекенду, MySQL як надійна СУБД, та React для побудови сучасного динамічного інтерфейсу користувача. Таким чином, перший розділ закладає концептуальну і технічну основу для подальшої розробки системи, враховуючи як функціональні потреби студії, так і технічні можливості реалізації.

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		20

РОЗДІЛ 2 ПРОЄКТУВАННЯ СИСТЕМИ АДМІНІСТРУВАННЯ

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

На цьому етапі проєктування було визначено ключові сценарії взаємодії користувачів із системою та сформовано загальне уявлення про її внутрішню структуру. В основі системи лежить об'єктно-орієнтований підхід, який дозволяє логічно впорядкувати дані й функціональність, що полегшує як реалізацію, так і подальшу підтримку.

У системі передбачено кілька типів користувачів — адміністратор і клієнт. Кожен із них має свій набір дій. Наприклад, адміністратор має змогу створювати розклад, додавати нові заняття, призначати інструкторів, переглядати фінансову статистику, редагувати інформацію про клієнтів і працівників. Клієнти, у свою чергу, записуються на заняття, оплачують їх, бачать майбутні уроки у своєму кабінеті та отримують нагадування.

У межах системи реалізуються такі основні сценарії: реєстрація та авторизація користувачів, створення та редагування занять, запис на уроки, керування фінансами, надсилання повідомлень, автоматичні розсилки, ведення статистики. Для цього створено відповідні сутності — такі як User, Lesson, Instructor, Client, Payment, Notification, Course, Studio тощо. Вони тісно пов'язані між собою. Наприклад, один клієнт може мати багато платежів, одне заняття — багато клієнтів, інструктор — багато уроків. Такі зв'язки дозволяють побудувати гнучку модель даних.

Весь код будується на принципах об'єктно-орієнтованого програмування. Це означає, що дані та методи, які з ними працюють, об'єднані в класи. Водночас застосовується наслідування (наприклад, Instructor і Client є розширенням базового класу User), а також інші ООП-принципи, як-от інкапсуляція та поліморфізм. Додатково використовуються сервісні класи для обробки логіки (наприклад, для надсилання повідомлень або обробки оплат), що дозволяє розділити відповідальність і зробити систему більш гнучкою.

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		21

У результаті створюється модель [10], де кожен компонент виконує чітко визначену роль, а взаємодія між ними відбувається через зрозумілі інтерфейси. Це дозволяє не лише реалізувати поточну функціональність, а й легко розширити систему в майбутньому — додати нові типи занять, ролі користувачів чи зовнішні сервіси. Деякі моделі системи представлені на рис. 2.1.

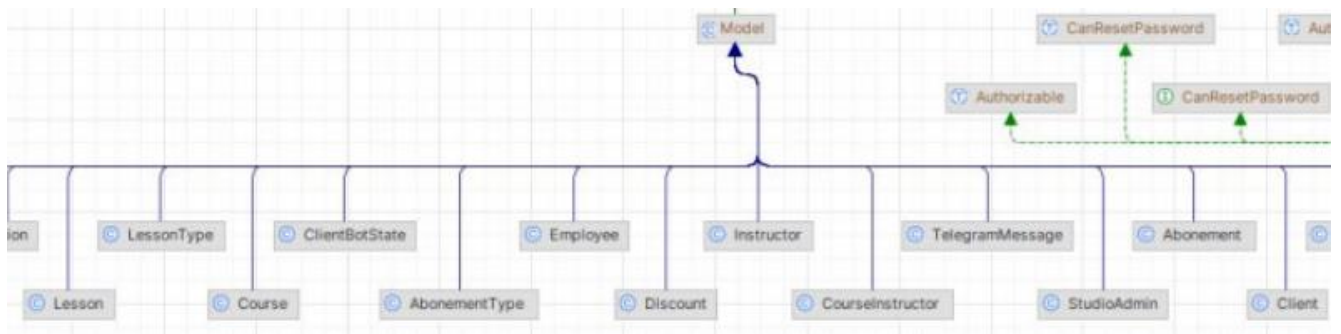


Рисунок 2.1 – Діаграма моделей системи адміністрування студії

Рис. 2.2 ілюструє загальну логіку роботи серверної частини системи та показує, як саме організовано обробку основних запитів користувача. У рамках платформи реалізовано розподіл функціоналу між окремими контролерами, кожен з яких відповідає за певну частину системи [3]. Це дозволяє підтримувати порядок у коді, уникати дублювання логіки та полегшує масштабування й обслуговування проєкту.

Наприклад, є окремий контролер для користувачів, через який здійснюється реєстрація, вхід до системи, редагування профілю. Інший контролер відповідає за уроки — він приймає запити на створення занять, їх редагування, видалення або перегляд. Також є контролери для клієнтів, які зберігають дані про людей, що записуються на заняття. Є фінансовий контролер, який відповідає за транзакції та звіти. Кожен із них взаємодіє з відповідними моделями та часто використовує певні сервіси, щоб не перевантажувати себе зайвими обчисленнями.

Уся ця структура дозволяє не лише ефективно обробляти запити, але й забезпечує гнучкість: за потреби можна легко додати новий функціонал, не зачіпаючи вже існуючий. Це особливо важливо для масштабованих систем, де з часом зростає як кількість функцій, так і навантаження.

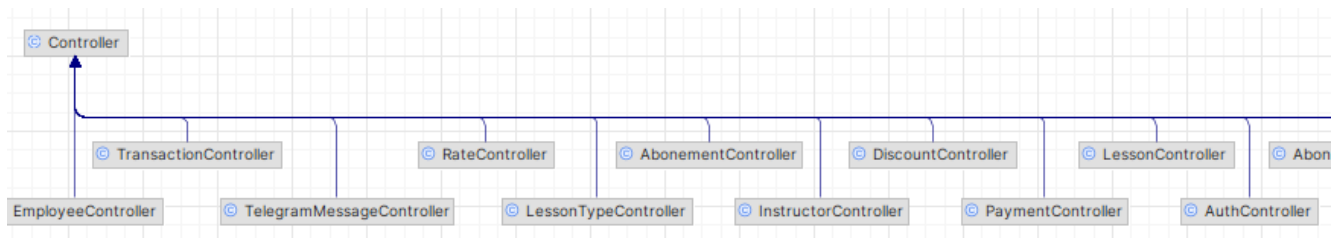


Рисунок 2.2 – Діаграма контролерів системи адміністрування студії

На рис. 2.3 представлено діаграму, що відображає ті елементи системи, які працюють у фоновому режимі та не потребують постійної участі адміністратора або викладача. Такі автоматизовані функції суттєво спрощують повсякденну роботу працівників студії, дозволяючи їм зосередитися на важливіших завданнях — комунікації з клієнтами, проведенні занять, аналізі результатів.

У системі передбачено надсилення автоматичних нагадувань про майбутні заняття, щоб клієнти не забували про свої записи. Також реалізовано розсилку привітань або сповіщень про зміни в розкладі, що дозволяє підтримувати контакт із клієнтом без потреби у ручному втручанні. Крім цього, система самостійно формує регулярні звіти про фінанси, відвідування чи інші важливі показники й надсилає їх адміністрації.

Автоматично працює й логіка оновлення розкладу — наприклад, коли інструктор змінює час проведення свого уроку, це одразу відображається у всіх пов'язаних модулях системи. Також працюють перевірки на помилки, наприклад, якщо два уроки накладаються один на одного, система може попередити про це ще до збереження.

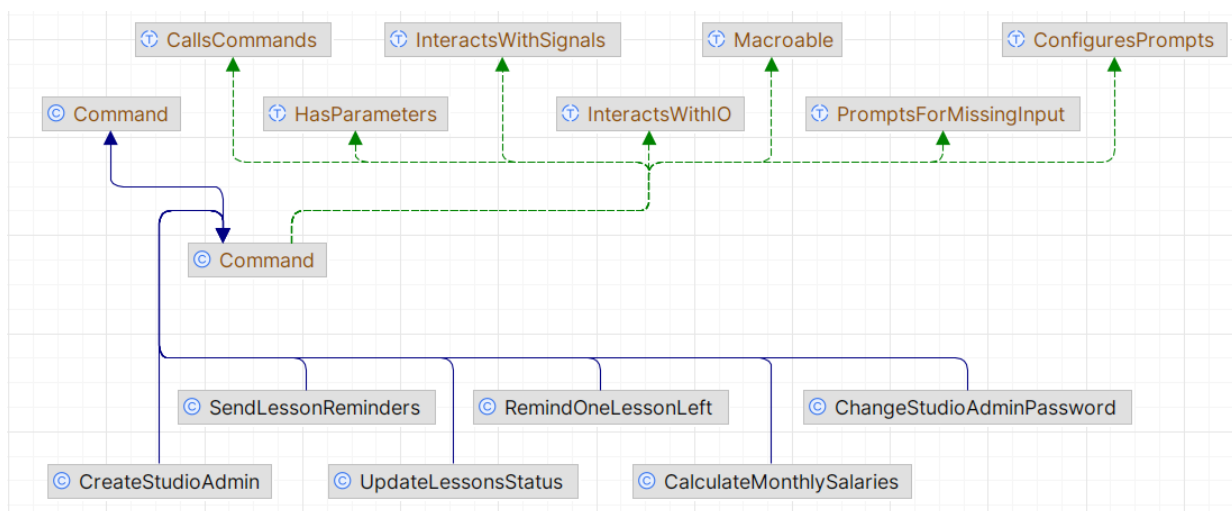


Рисунок 2.3 – Діаграма автоматичних функцій системи адміністрування студії

Загалом автоматизація охоплює ті зони, які зазвичай забирають багато часу при ручному виконанні, і дозволяє зробити роботу студії набагато ефективнішою та сучаснішою.

2.2 Розробка бази даних системи

Проектування бази даних [26] є ключовим етапом при створенні інформаційної системи, оскільки саме база відповідає за цілісність, зв'язки та доступність даних. У межах системи адміністрування навчальної студії база даних реалізована у вигляді реляційної моделі з чітко визначеними таблицями, зв'язками та типами атрибутів.

У центрі схеми перебуває таблиця lessons, яка фіксує всю інформацію про заняття: тип заняття, курс, інструктора, студію, час початку та закінчення, кількість клієнтів, ціну та статус. Вона пов'язана з багатьма іншими таблицями, зокрема lesson_types, courses, studios, instructors, а також з таблицею clients, які беруть участь у заняттях.

Окремо реалізовано таблицю transactions, що забезпечує облік фінансових операцій: оплати, витрати, знижки. Кожна транзакція має посилання на клієнта, інструктора, адміністратора, тип транзакції, пов'язаний абонемент, а також поле для опису. Для знижок існує окрема таблиця discounts.

Система підтримує роботу з абонементом через таблиці abonements та abonement_types, які описують параметри підписок клієнтів, тривалість, ціну, зв'язок із курсами. Додатково реалізовано таблицю abonement_type_rates, яка дозволяє налаштовувати різні ставки для абонементів залежно від типу заняття, інструктора та курсу.

Інструктори мають окрему таблицю instructors, яка пов'язана з таблицею employees, що містить загальну інформацію про працівників: ім'я, посаду, телефон, зарплату. Зарплати деталізуються через таблицю salaries, яка дозволяє зберігати різні параметри компенсацій.

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		24

Для аналітики використовується таблиця rates, яка дозволяє зафіксувати різні ставки для комбінацій курсу, типу заняття та інструктора. Це забезпечує гнучке ціноутворення та підрахунок виплат.

Для управління адміністративним доступом реалізовані таблиці studio_admin, personal_access_tokens, password_reset_tokens, які дозволяють керувати сесіями, паролями та доступом через API.

Клієнти зберігаються в таблиці clients, яка містить всю необхідну контактну інформацію: Telegram, Instagram, номер телефону, дату народження, контакт у разі надзвичайної ситуації. Для інтеграції з Telegram передбачено таблиці telegram_messages та client_bot_states, які фіксують повідомлення, статуси взаємодії з ботом і стан чату.

Система також підтримує зв'язки між курсами та інструкторами (course_instructor), а також дозволяє керувати штатним розподілом інструкторів по студіях та курсах. Усі зв'язки реалізовано через зовнішні ключі, що забезпечує цілісність даних.

Таким чином, база даних нараховує 18 таблиць: clients, employees, instructors, courses, studios, lesson_types, lessons, transactions, abonements, rates, discounts, salaries, course_instructors, abonement_type_rates, abonement_types, telegram_messages, studio_admins, client_bot_states.

Структура бази даних розроблена з урахуванням можливого масштабування та зручної підтримки при додаванні нових функцій системи у майбутньому. Вона передбачає логічну організацію даних і зв'язків між ними, що забезпечує гнучкість та адаптивність до змін бізнес-процесів. Наразі база даних є потужним інструментом для ефективного зберігання й обробки інформації, необхідної для керування всіма аспектами роботи студії — від занять і клієнтів до оплати, аналітики та автоматизації процесів.

На рис. 2.4 подано діаграму бази даних, яка відображає логічну модель з основними сутностями та їхніми взаємозв'язками.

Опис кожної з таблиць бази даних, а саме назви полів, тип даних, що зберігається та їх призначення наведений у табл. 2.1 -2.18.

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		25

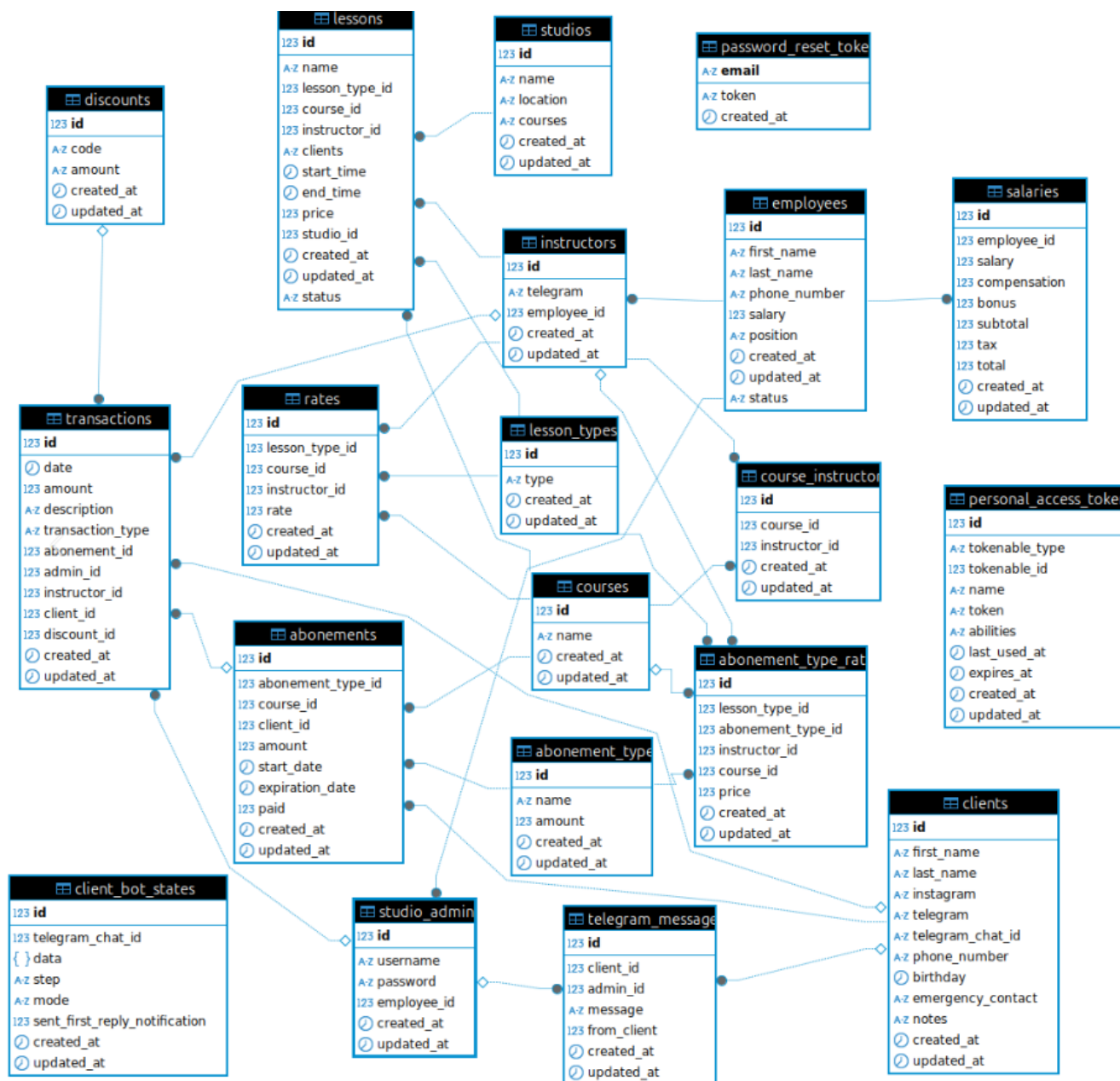


Рисунок 2.4 – Діаграма бази даних

Таблиця 2.1

Опис таблиці clients

Поле	Тип	Призначення
1	2	3
id	integer	Унікальний ідентифікатор клієнта
first_name	string	Ім'я клієнта
last_name	string	Прізвище клієнта
instagram	string	Instagram-нік клієнта

1	2	3
telegram	string	Telegram-нік клієнта
telegram_chat_id	bigint	Унікальний ID чату Telegram
phone_number	string	Номер телефону клієнта
birthday	date	Дата народження клієнта
emergency_contact	string	Контактна особа у випадку надзвичайної ситуації
notes	text	Додаткові нотатки
created_at	timestamp	Дата створення запису
updated_at	timestamp	Дата останнього оновлення запису

Таблиця 2.2

Опис таблиці employees

Поле	Тип	Призначення
id	integer	Унікальний ідентифікатор працівника
first_name	string	Ім'я працівника
last_name	string	Прізвище працівника
phone_number	string	Номер телефону
salary	decimal	Заробітна плата
position	string	Посада працівника
created_at	timestamp	Дата створення запису
updated_at	timestamp	Дата останнього оновлення запису
status	boolean	Статус активності (активний/неактивний)

Таблиця 2.3

Опис таблиці instructors

Поле	Тип	Призначення
id	integer	Унікальний ідентифікатор інструктора
telegram	string	Telegram-нік інструктора
employee_id	integer	Зовнішній ключ на таблицю employees
created_at	timestamp	Дата створення запису
updated_at	timestamp	Дата останнього оновлення запису

Таблиця 2.4

Опис таблиці courses

Поле	Тип	Призначення
id	integer	Унікальний ідентифікатор курсу
name	string	Назва курсу
created_at	timestamp	Дата створення запису
updated_at	timestamp	Дата останнього оновлення запису

Таблиця 2.5

Опис таблиці studios

Поле	Тип	Призначення
id	integer	Унікальний ідентифікатор студії
name	string	Назва студії
location	string	Місцезнаходження студії
created_at	timestamp	Дата створення запису
updated_at	timestamp	Дата останнього оновлення запису

Таблиця 2.6

Опис таблиці lesson_types

Поле	Тип	Призначення
id	integer	Унікальний ідентифікатор типу заняття
type	string	Назва типу заняття
created_at	timestamp	Дата створення запису
updated_at	timestamp	Дата останнього оновлення запису

Таблиця 2.7

Опис таблиці lessons

Поле	Тип	Призначення
1	2	3
id	integer	Унікальний ідентифікатор заняття
course_id	integer	Зовнішній ключ на таблицю courses
instructor_id	integer	Зовнішній ключ на таблицю instructors
studio_id	integer	Зовнішній ключ на таблицю studios
lesson_type_id	integer	Зовнішній ключ на таблицю lesson_types

1	2	3
date	date	Дата проведення заняття
start_time	time	Час початку заняття
end_time	time	Час закінчення заняття
created_at	timestamp	Дата створення запису
updated_at	timestamp	Дата останнього оновлення запису

Таблиця 2.8

Опис таблиці transactions

Поле	Тип	Призначення
id	integer	Унікальний ідентифікатор транзакції
amount	decimal	Сума транзакції
purpose	string	Призначення платежу
method	string	Спосіб оплати (готівка, картка, онлайн тощо)
date	date	Дата транзакції
admin_id	integer	Зовнішній ключ на адміністратора (таблиця employees)
client_id	integer	Зовнішній ключ на клієнта (таблиця clients)
instructor_id	integer	Зовнішній ключ на інструктора (таблиця instructors)
created_at	timestamp	Дата створення запису
updated_at	timestamp	Дата останнього оновлення запису

Таблиця 2.9

Опис таблиці abonements

Поле	Тип	Призначення
id	integer	Унікальний ідентифікатор абонементу
client_id	integer	Зовнішній ключ на клієнта
course_id	integer	Зовнішній ключ на курс
lesson_type_id	integer	Зовнішній ключ на тип заняття
lessons_count	integer	Кількість занять у межах абонементу
price	decimal	Вартість абонементу
is_active	boolean	Ознака активності абонементу
created_at	timestamp	Дата створення запису
updated_at	timestamp	Дата останнього оновлення запису

Таблиця 2.10

Опис таблиці rates

Поле	Тип	Призначення
id	integer	Унікальний ідентифікатор ставки
instructor_id	integer	Зовнішній ключ на інструктора
course_id	integer	Зовнішній ключ на курс
lesson_type_id	integer	Зовнішній ключ на тип заняття
rate	decimal	Ставка оплати за одне заняття
created_at	timestamp	Дата створення запису
updated_at	timestamp	Дата останнього оновлення запису

Таблиця 2.11

Опис таблиці discounts

Поле	Тип	Призначення
id	integer	Унікальний ідентифікатор ставки
code	string	Код знижки, що може вводиться клієнтом
amount	integer	Розмір знижки у відсотках
created_at	timestamp	Дата створення запису
updated_at	timestamp	Дата останнього оновлення запису

Таблиця 2.12

Опис таблиці salaries

Поле	Тип	Призначення
1	2	3
id	integer	Унікальний ідентифікатор ставки
Employee_id	integer	Ідентифікатор працівника, для якого розраховується зарплата
Salary	decimal	Базова ставка зарплати працівника за певний період
Compensation	decimal	Додаткові компенсаційні виплати (за проведені заняття, тощо)
bonus	decimal	Премії або додаткові заохочення, які нараховуються окремо від основної зарплати
Subtotal	decimal	Проміжна сума, що є підсумком поля salary, compensation та bonus перед оподаткуванням

1	2	3
Tax	decimal	Сума податків, яка вираховується із загальної суми нарахувань.
total	decimal	Фінальна сума до сплати з врахуванням податків.
created_at	timestamp	Дата створення запису
updated_at	timestamp	Дата останнього оновлення запису

Таблиця 2.13

Опис таблиці abonement_type_rates

Поле	Тип	Призначення
id	integer	Унікальний ідентифікатор ставки
instructor_id	integer	Зовнішній ключ на інструктора
course_id	integer	Зовнішній ключ на курс
lesson_type_id	integer	Зовнішній ключ на тип заняття
abonement_type_id	integer	Зовнішній ключ на тип абонементу
price	decimal	Вартість абонементу
created_at	timestamp	Дата створення запису
updated_at	timestamp	Дата останнього оновлення запису

Таблиця 2.14

Опис таблиці telegram_messages

Поле	Тип	Призначення
id	integer	Унікальний ідентифікатор ставки
client_id	integer	Зовнішній ключ, що вказує на клієнта
admin_id	integer	Зовнішній ключ на адміністратора
message	text	Текст повідомлення, що було надіслано або отримано
from_client	boolean	Логічне поле, що визначає, ким було надіслано повідомлення: true — клієнтом, false — адміністратором.
created_at	timestamp	Дата створення запису
updated_at	timestamp	Дата останнього оновлення запису

Таблиця 2.15

Опис таблиці client_bot_states

Поле	Тип	Призначення
id	integer	Унікальний ідентифікатор ставки
telegram_chat_id	bigint	Ідентифікатор Telegram-чату клієнта
data	json	Додаткові тимчасові дані або параметри, які зберігаються під час діалогу з ботом.
step	string	Поточний крок у сценарії взаємодії клієнта з ботом
mode	string	Режим роботи бота, у якому перебуває клієнт
sent_first_reply_notification	boolean	Показник, чи було надіслано перше автоматичне повідомлення
created_at	timestamp	Дата створення запису
updated_at	timestamp	Дата останнього оновлення запису

Таблиця 2.16

Опис таблиці abonement_types

Поле	Тип	Призначення
id	integer	Унікальний ідентифікатор ставки
name	string	Назва абонементу
amount	integer	Кількість занять у абонементі
created_at	timestamp	Дата створення запису
updated_at	timestamp	Дата останнього оновлення запису

Таблиця 2.17

Опис таблиці course_instructors

Поле	Тип	Призначення
id	integer	Унікальний ідентифікатор ставки
instructor_id	integer	Зовнішній ключ на інструктора
course_id	integer	Зовнішній ключ на курс
created_at	timestamp	Дата створення запису
updated_at	timestamp	Дата останнього оновлення запису

Опис таблиці studio_admins

Поле	Тип	Призначення
id	integer	Унікальний ідентифікатор ставки
employee_id	integer	Зовнішній ключ на інструктора
Username	string	Унікальне ім'я користувача, для входу в систему
password	string	Хешований пароль для доступу до адміністративної панелі
created_at	timestamp	Дата створення запису
updated_at	timestamp	Дата останнього оновлення запису

2.3 Проєктування та реалізація алгоритмів роботи системи

Уся система побудована з урахуванням принципів зручності та інтуїтивної зрозумілості для користувачів. Основна мета — зробити щоденну роботу студії максимально простою та ефективною. Інтерфейс і логіка системи дозволяють швидко орієнтуватися в її функціоналі навіть новим користувачам. Завдяки продуманій навігації та логічній структурі розділів, взаємодія з системою не потребує додаткового навчання.

У першу чергу, працювати з нею починає адміністратор студії. Після авторизації за допомогою логіна й пароля, він потрапляє до адміністративної панелі. Тут доступний повний набір інструментів для керування усіма аспектами діяльності: адміністратор може створювати нові уроки, редагувати або видаляти наявні, призначати або змінювати розклад, а також бачити всі актуальні події в системі.

Окрім цього, адміністратор має можливість додавати нових працівників до системи, призначати їм відповідні ролі та повноваження, а також контролювати їхню діяльність. Через панель керування він також оперує даними про клієнтів, обробляє заявки на запис, створює та редагує категорії курсів, призначає заняття в конкретні зали, що дозволяє ефективно планувати ресурси студії.

Для зручності клієнтів реалізовано Telegram-бот, який виступає простим і швидким інструментом взаємодії зі студією. Якщо клієнт хоче скористатися

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		33

послугами, він заходить у бот, де йому доступний список усіх активних курсів. Клієнт може переглядати розклад занять, обирати відповідні курси та здійснювати запис. У боті також є можливість переглядати власні записи та актуальну інформацію щодо занять (рис. 2.5). Також він може купити абонемент (рис. 2.6).

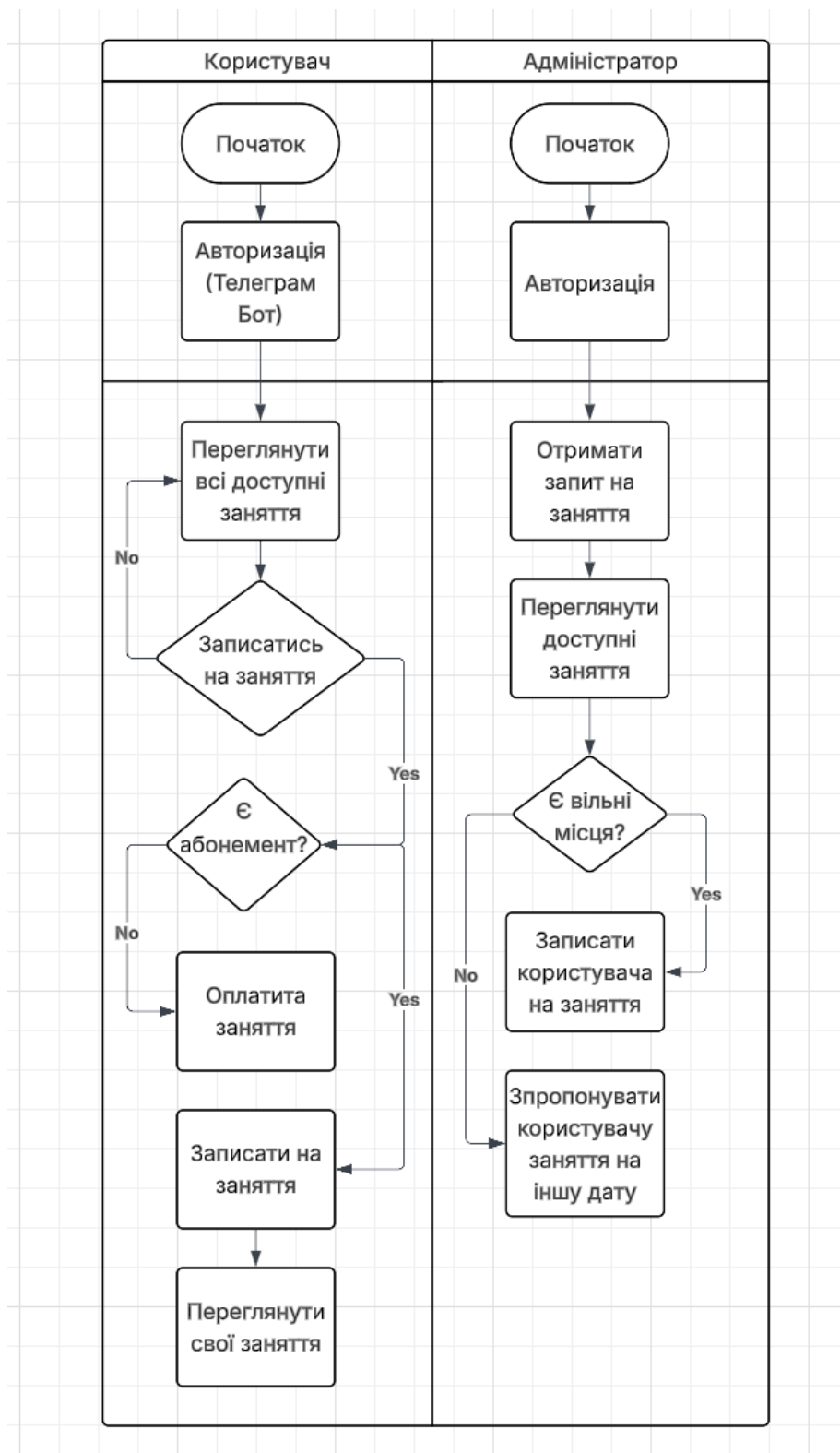


Рисунок 2.5 – Діаграма активності запису на заняття

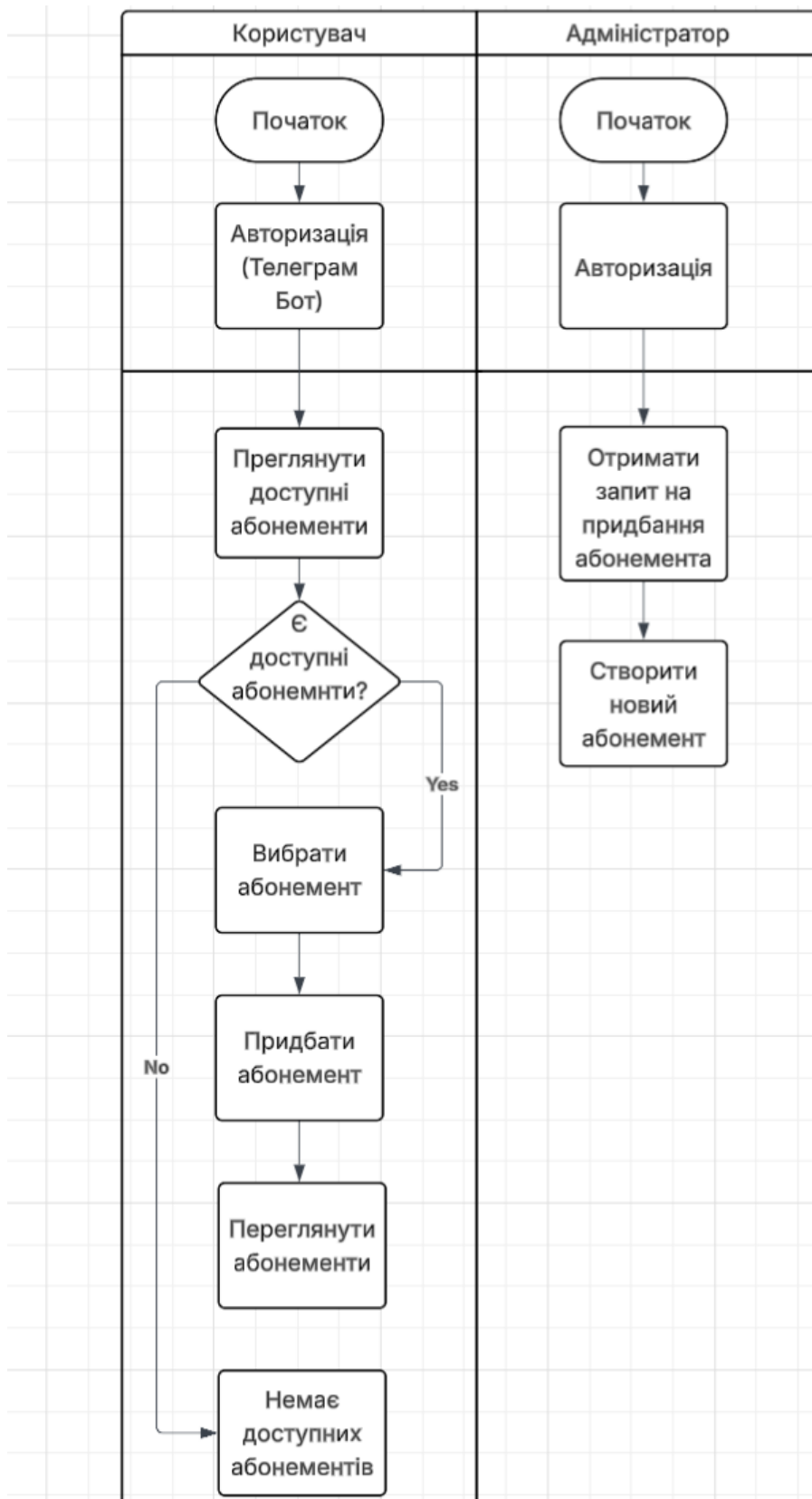


Рисунок 2.6 – Діаграма активності купівлі/додавання абонементу

Усю оплату клієнти зазвичай здійснюють напряму, або через адміністратора, що потім вносить ці дані в систему. У розділі «Фінанси» адміністратор може бачити список усіх транзакцій, додавати нові, редагувати чи видаляти, а також друкувати

фінансові звіти. Є фільтри за датою, сумою та способом оплати — це дозволяє швидко знаходити потрібну інформацію.

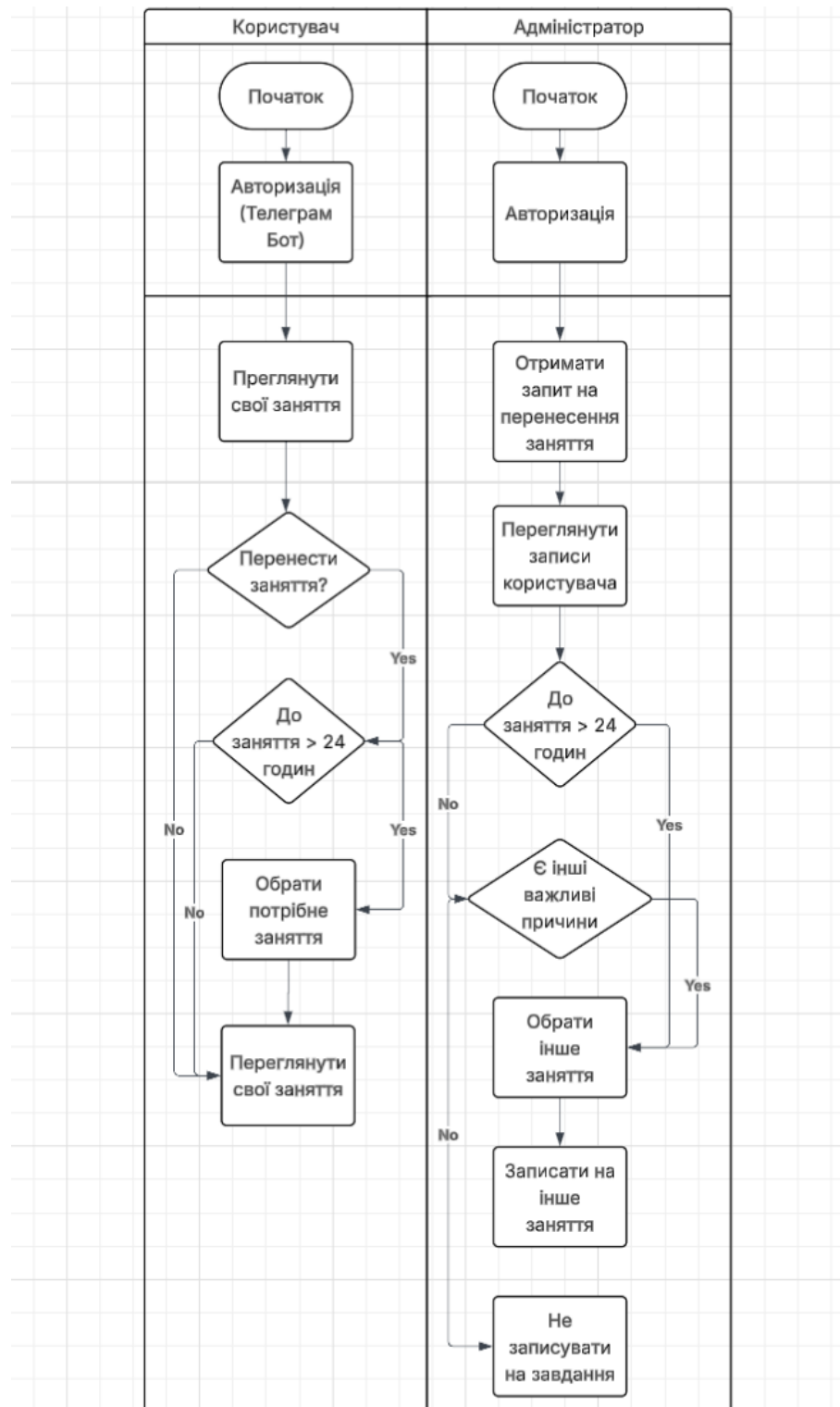


Рисунок 2.8 – Діаграма активності перенесення занять

Уся система працює просто і зрозуміло, щоб адміністратор мав під рукою всі інструменти, а клієнти могли зручно подавати заявки через Telegram, не заходячи

в окремий сайт чи застосунок. Наприклад перенести заняття (рис. 2.8) або зв'язатися з адміністратором (рис. 2.9).

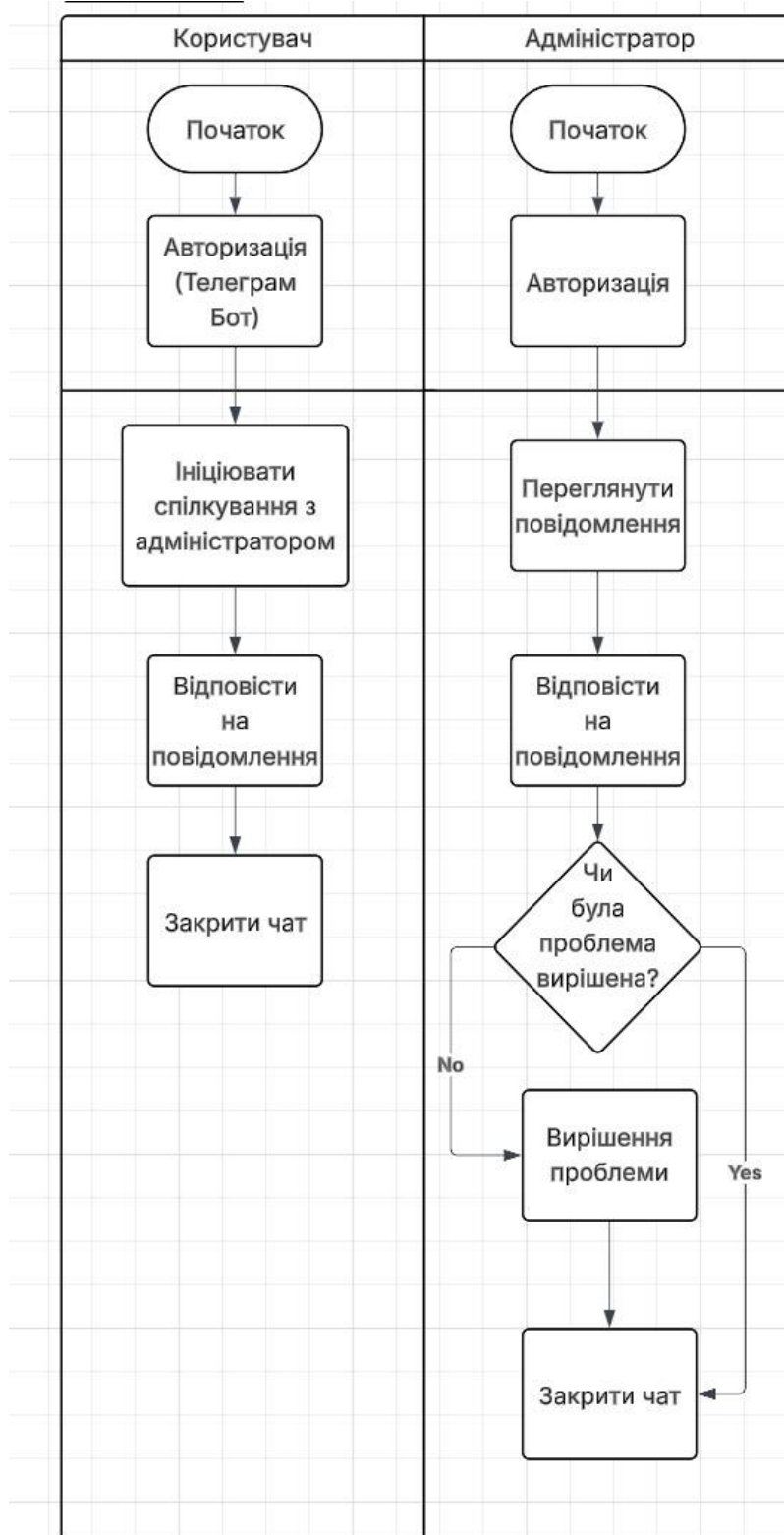


Рисунок 2.9 - Діаграма активності спілкування через систему

Для взаємодії компонентів проекту «Система адміністрації навчальної студії» було прийнято рішення використати низку популярних дизайн-патернів, які

забезпечать структуровану архітектуру [19], спростять підтримку, масштабування та модернізацію системи. Основною архітектурною основою став шаблон MVC [24], який дозволяє чітко розділити логіку програми на окремі частини: дані (моделі), взаємодію з користувачем (контролери) і відображення (у випадку, коли використовується web-інтерфейс). Це спрощує подальші зміни у проєкті, адже одна частина коду не плутається з іншою.

Також застосовувався патерн репозиторію, завдяки якому вся логіка взаємодії з базою даних виноситься в окремі класи. Це робить контролери чистішими і дозволяє легше тестувати частини системи. Наприклад, запити до таблиць занять або клієнтів не пишуться безпосередньо в контролерах, а винесені в репозиторії.

Для того, щоб певні сервіси в системі створювалися лише один раз і не дублювалися по всьому застосунку, використовувався патерн Singleton. Зокрема, це стосується Telegram-бота, який ініціалізується один раз і через який проходить вся взаємодія з клієнтами. Завдяки цьому не виникає конфліктів або повторного створення з'єднань із зовнішнім API.

Ще одним корисним підходом став Strategy — коли потрібно обирати певну поведінку динамічно. Наприклад, для фільтрації фінансових транзакцій або побудови розкладу може використовуватися різна стратегія, яка підставляється в залежності від вибору користувача або налаштувань системи.

2.4 Реалізація функціоналу системи

Розробка системи ведеться з урахуванням сучасних підходів до архітектури вебзастосунків, зокрема з активним використанням контейнеризації за допомогою Docker. Це рішення дозволяє створити стабільне, контрольоване середовище як для розробки, так і для подальшого розгортання проєкту на сервері.

Проєкт складається з кількох взаємозалежних компонентів, серед яких: серверна частина (на Laravel), база даних (MySQL), клієнтська частина (React), а також Telegram-бот, що забезпечує інтерактивну взаємодію з користувачами. Кожен із цих компонентів ізольовано в окремому контейнері, що дозволяє уникати

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		38

конфліктів залежностей, прискорити запуск проєкту на будь-якій машині, незалежно від ОС, та забезпечити повторюваність результату.

Використання Docker [6] дозволяє точно фіксувати версії всього необхідного програмного забезпечення (PHP, Node.js, MySQL тощо) у спеціальних конфігураційних файлах (Dockerfile, docker-compose.yml). Таким чином, будь-який розробник може запустити повноцінне середовище командою `docker compose up`, без складних налаштувань локально.

Окрім зручності в розробці, Docker також надає великі переваги при CI/CD — автоматизованому розгортанні коду. Завдяки цьому можна створити скрипти, які запускають тести, оновлюють базу, роблять резервні копії або деплоять нову версію системи. Усе це підвищує надійність і зменшує ризики при оновленнях.

З розвитком проєкту Docker дозволяє масштабувати систему, наприклад: запускати окремі контейнери для аналітики, адміністративного моніторингу, створювати staging-версії (тестові копії продакшну), а також розгортати нові сервіси без втручання в основну інфраструктуру. У цьому проєкті реалізовані такі контейнери.

Контейнер для Laravel API (бекенд): містить образ PHP з установленим Composer та Laravel. Налаштування виконуються через .env-файл, де задаються параметри доступу до бази даних, Redis, портів тощо. Підключено volume для збереження коду та синхронізації змін у реальному часі. Підключення контейнеру у докерфайл зображено на рис. 2.10.

Контейнер для MySQL (база даних): містить образ із попередньо налаштованою СУБД. Ініціалізація бази даних виконується автоматично при першому запуску. Підтримується можливість створення дампів і відновлення з резервних копій.

Контейнер для фронтенду (React): у даній системі використовується контейнер з Nginx як статичний сервер для роздачі зібраних файлів React-застосунку. У процесі розробки застосунків спочатку збирається за допомогою Node.js, після чого згенеровані статичні файли копіюються до директорії, яку обслуговує Nginx. Такий підхід забезпечує високу продуктивність при розгортанні

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		39

та зручність в обслуговуванні інтерфейсу. У режимі розробки може бути також налаштований live reload через окреме середовище, але для продакшн-версії використовується саме Nginx.

```
version : '3.8'
services:
  backend:
    build:
      context: ./backend
      dockerfile: Dockerfile
    environment:
      DB_HOST: db
      DB_DATABASE: studio_db
      DB_USERNAME: root
      DB_PASSWORD: root
    extra_hosts:
      - 'host.docker.internal:host-gateway'
    container_name: backend
    volumes:
      - ./backend:/app/backend
    depends_on:
      - db
    networks:
      - app
```

Рисунок 2.10 – Частина докерфайлу для створення Laravel контейнеру

Бекенд системи було реалізовано з використанням фреймворку Laravel версії 10, який є одним з найпопулярніших PHP-фреймворків для створення веб-застосунків та REST API [20]. Laravel надає зручну структуру, розширюваність та підтримку великої кількості інструментів, що значно спрощує розробку. У даній системі він використовується для обробки всіх запитів, пов'язаних із ключовими об'єктами – користувачами, працівниками, клієнтами, курсами, уроками та фінансовими транзакціями.

Усі операції над цими сутностями реалізовані за допомогою контролерів та моделей. Кожен контролер відповідає за певну частину логіки – наприклад, LessonController (Додаток А) обробляє створення, редагування та видалення уроків, а EmployeeController (Додаток Б) – роботу з працівниками. Вони базуються на механізмах Laravel API Resource Controllers, що дозволяє організовано структурувати логіку CRUD (створення, читання, оновлення, видалення).

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
						40
Змн.	Арк.	№ докум.	Підпис	Дата		

Коли користувач взаємодіє з клієнтським інтерфейсом (наприклад, адміністратор студії створює новий урок чи транзакцію), формується HTTP-запит. Цей запит надсилається на сервер на одну з попередньо визначених адрес: /api/lessons, /api/employees, /api/transactions тощо.

У відповідному контролері запит проходить валідацію – перевірку даних на коректність. Наприклад, при створенні уроку перевіряється, чи заповнено назву, чи вказано допустимі дати початку та завершення, чи існує інструктор з вказаним ID. Після успішної валідації дані зберігаються у базі даних за допомогою моделі.

Таким чином, фреймворк автоматично перевіряє вхідні дані та дозволяє зручно передавати їх до моделі, де вони перетворюються у новий запис у базі даних.

Окрім створення, реалізовано також функції оновлення та видалення записів. Для цього в кожному контролері реалізовані методи update та destroy. Наприклад, урок можна відредагувати або видалити за допомогою відповідного PUT чи DELETE-запиту.

Система також передбачає можливість фільтрації даних. Це особливо важливо для сторінок, що містять велику кількість записів, як-от фінансові транзакції. Користувач має змогу відібрати записи за різними критеріями: датою, сумою, способом оплати, адміністратором, клієнтом чи інструктором. Для реалізації цієї логіки на бекенді створюються динамічні SQL-запити, які адаптуються до введених фільтрів. Це досягається шляхом побудови умов where на основі вхідного запиту користувача.

Такий підхід дозволяє створити гнучку, розширювану систему з чітким розділенням відповідальностей. Laravel забезпечує високу продуктивність і зручну роботу з базою даних, тоді як REST API дозволяє ефективно обмінюватися даними з клієнтською частиною, розробленою на React.

Фронтенд цієї системи побудований на базі JavaScript-бібліотеки React [15], яка дозволяє створювати швидкий та інтерактивний інтерфейс. Завдяки React сторінка не перезавантажується при кожній дії – усе оновлюється динамічно, що робить роботу зі системою швидкою та зручною.

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		41

Усі основні дії в інтерфейсі побудовані так, щоби користувач міг працювати максимально інтуїтивно. Наприклад, на сторінці з розкладом занять (Додаток В) можна бачити тижневу сітку уроків. Щоб створити новий урок, не потрібно переходити на іншу сторінку. Достатньо натиснути на порожню клітинку в розкладі — відкриється модальне вікно, де можна заповнити всі потрібні дані. Це ж вікно використовується і для редагування існуючих уроків, якщо натиснути на вже створене заняття. Так само легко можна і видалити його, якщо потрібно.

Крім уроків, інтерфейс дозволяє керувати працівниками, клієнтами та фінансовими транзакціями. Є окремі розділи, де можна додавати нових працівників, редагувати їхні дані, або створювати фінансові записи.

Уся взаємодія з бекендом відбувається через бібліотеку Axios [2], яка виконує HTTP-запити. Ці запити автоматично відправляються, коли сторінка завантажується або коли змінюються якісь дані, тому користувач завжди бачить актуальну інформацію.

Форма для створення або редагування уроку зроблена у вигляді модального вікна, яке з'являється поверх основної сторінки. Усі поля у формі (наприклад, назва, інструктор, час початку і завершення) пов'язані з локальним станом компонента, тобто вони одразу реагують на введення користувача. Коли натискається кнопка "Зберегти", дані надсилаються на сервер через POST або PUT-запит, залежно від того, чи це новий урок, чи редагування існуючого.

Завдяки такому підходу інтерфейс працює швидко, логічно та зручно – усе, що потрібно, завжди під рукою.

У проєкті як система управління базами даних використовується MySQL – надійне та швидке рішення, яке чудово підходить для веб-додатків [28]. Уся структура бази чітко відповідає логіці самої системи: є таблиці для користувачів, працівників, клієнтів, уроків, транзакцій, курсів, ставок інструкторів тощо. Кожна таблиця відповідає певній "сутності" — умовно кажучи, частині реального світу, з якою працює система.

Оскільки проєкт реалізований на Laravel, для створення структури бази використовуються міграції. Це зручно тим, що структура бази описується прямо у

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		42

кодів: можна швидко внести зміни, додати зв'язки між таблицями або змінити тип поля, і все це буде контролювано та повторювано на будь-якому сервері.

Зв'язки між таблицями реалізовані через зовнішні ключі — тобто, наприклад, кожен урок у базі має посилання на конкретного інструктора або клієнта. І якщо видалити інструктора, автоматично видаляться й пов'язані з ним записи, якщо так вказано в міграції (через `onDelete('cascade')`). Це дозволяє тримати базу в порядку й уникати "висячих" записів.

Але на практиці розробник рідко працює безпосередньо з SQL-запитами. Замість цього в Laravel використовують Eloquent ORM — потужний інструмент, який дозволяє працювати з базою у вигляді об'єктів. Наприклад, можна написати `$lesson->instructor->name`, і одразу отримати ім'я інструктора, пов'язаного з уроком, без жодного вручну написаного JOIN-запиту.

Всі ці зв'язки — один до одного, один до багатьох, багато до багатьох — описуються безпосередньо в моделях, і завдяки цьому можна дуже зручно будувати запити, фільтрувати дані, оновлювати або видаляти записи, не заглиблюючись у складність SQL.

Загалом, завдяки комбінації MySQL, Laravel-міграцій та Eloquent ORM, робота з даними у цьому проєкті побудована логічно, прозоро та зручно як для розробника, так і для підтримки системи в майбутньому.

Висновки до другого розділу

У процесі проєктування платформи для управління студією було проаналізовано всі основні сценарії використання системи відповідно до функціональних вимог. На основі цього аналізу було розроблено загальну архітектуру клієнтської та серверної частин.

Проєктування охоплювало побудову структури компонентів, що відповідають за обробку користувачів, працівників, клієнтів, уроків, фінансових транзакцій, курсів та ставок інструкторів. Було створено REST API на основі Laravel, яке забезпечує всі бізнес-операції, включаючи створення, редагування, видалення та фільтрацію даних. Усі запити проходять через контролери, які

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		43

здійснюють валідацію та взаємодіють з моделями Eloquent ORM для роботи з базою даних.

Зі сторони клієнта було реалізовано сучасний інтерфейс на React. Забезпечено зручну взаємодію користувача з системою: перегляд розкладу занять у вигляді сітки з підтримкою drag-and-drop, створення уроків натисканням на вільну клітинку, редагування й видалення записів, а також керування працівниками, клієнтами та фінансовими операціями.

Для зберігання даних використовується СУБД MySQL. Усі сутності системи зв'язані між собою за допомогою зовнішніх ключів, що описані в міграціях Laravel. Взаємозв'язки дублюються в Eloquent-моделях, що дозволяє легко працювати з даними через ORM-запити.

Таким чином, у другому розділі було реалізовано повноцінне проєктування клієнтської та серверної частин системи, розроблено структуру бази даних, описано механізми взаємодії між компонентами та забезпечено логіку основних дій користувачів у системі.

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		44

РОЗДІЛ 3 ІНТЕРФЕЙС І ПОРЯДОК РОБОТИ З СИСТЕМОЮ АДМІНІСТРУВАННЯ НАВЧАЛЬНОЇ СТУДІЇ

3.1 Порядок встановлення та налаштування параметрів системи

Встановлення та налаштування системи адміністрування навчальної студії передбачає розгортання повноцінної інфраструктури на основі Docker-контейнерів. Такий підхід дозволяє забезпечити ізоляцію середовищ, уникати конфліктів між залежностями, а також значно спрощує процес розгортання, масштабування та супроводу системи. Кожен компонент запускається у власному контейнері, що дозволяє гнучко керувати конфігурацією та оновленням окремих частин проєкту без впливу на інші частини.

До основних компонентів системи входять такі сервіси:

- Laravel Backend — API-сервер, реалізований на фреймворку Laravel, відповідає за взаємодію з базою даних, обробку логіки застосунку, авторизацію користувачів, обробку запитів з фронтенду та реалізацію бізнес-процесів.
- MySQL — система керування базами даних, яка зберігає всі необхідні дані: інформацію про клієнтів, заняття, працівників, транзакції, курси та інші об'єкти доменної області.
- React Frontend — сучасний клієнтський інтерфейс, розроблений за допомогою бібліотеки React, який забезпечує зручну взаємодію користувача з платформою, візуалізацію даних і динамічне оновлення сторінок.
- Nginx — продуктивний та стабільний веб-сервер, який використовується як зворотний проксі для маршрутизації запитів до фронтенду та API. Він також обслуговує статичні файли React-застосунку у продакшн-середовищі.

Для запуску всієї системи достатньо виконати команду `docker-compose up -d`, попередньо сконфігурувавши файл `.env`, який знаходиться в кореневій директорії проєкту. У цьому файлі вказуються всі необхідні змінні середовища, включаючи

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		45

параметри підключення до бази даних, секретні ключі, URL-адреси зовнішніх сервісів тощо. Без правильно заповненого .env запуск буде неможливим або некоректним.

Після успішного запуску система буде доступна у браузері за локальною адресою, наприклад, `http://localhost`, де відкриється головна сторінка інтерфейсу користувача. API, у свою чергу, буде доступне за адресою `http://localhost/api`, і саме до нього надходитимуть запити з фронтенду або зовнішніх інтеграцій. Така структура дозволяє гнучко розділяти логіку бекенду й інтерфейс користувача, забезпечуючи гнучкість і масштабованість системи в майбутньому.

У разі потреби система може бути легко перенесена на віддалений сервер або хмарну платформу без суттєвих змін у конфігурації. Усі оновлення й підтримка здійснюються централізовано через зміну окремих сервісів, що робить систему зручною для супроводу та розширення.

3.2 Структура інтерфейсу

Взаємодія користувачів (адміністраторів) відбувається через вебдодаток [23]. Головною сторінкою є сторінка розкладу, де користувач може переглянути розклад на поточний тиждень або на попередній чи наступний тиждень (рис. 3.1).

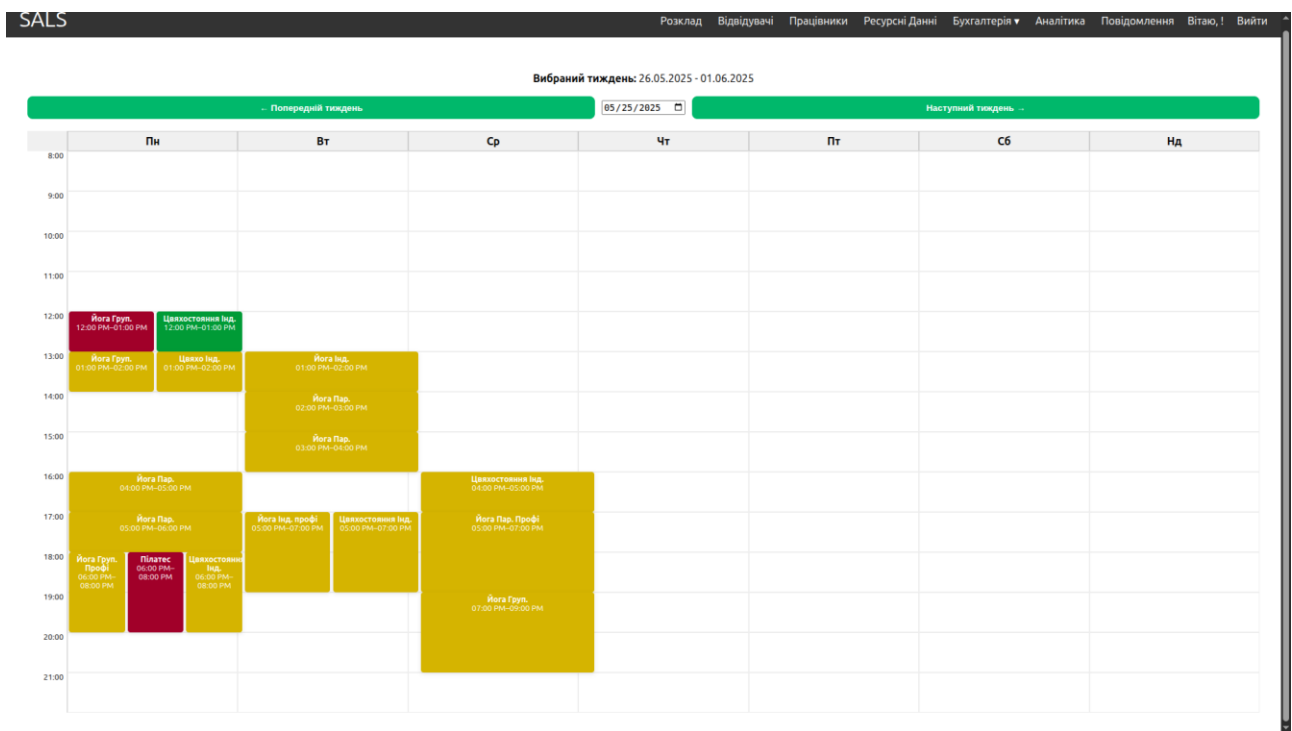


Рисунок 3.1 – Головна сторінка застосунку

Також користувач може переглянути деталі існуючого уроку (рис. 3.2).

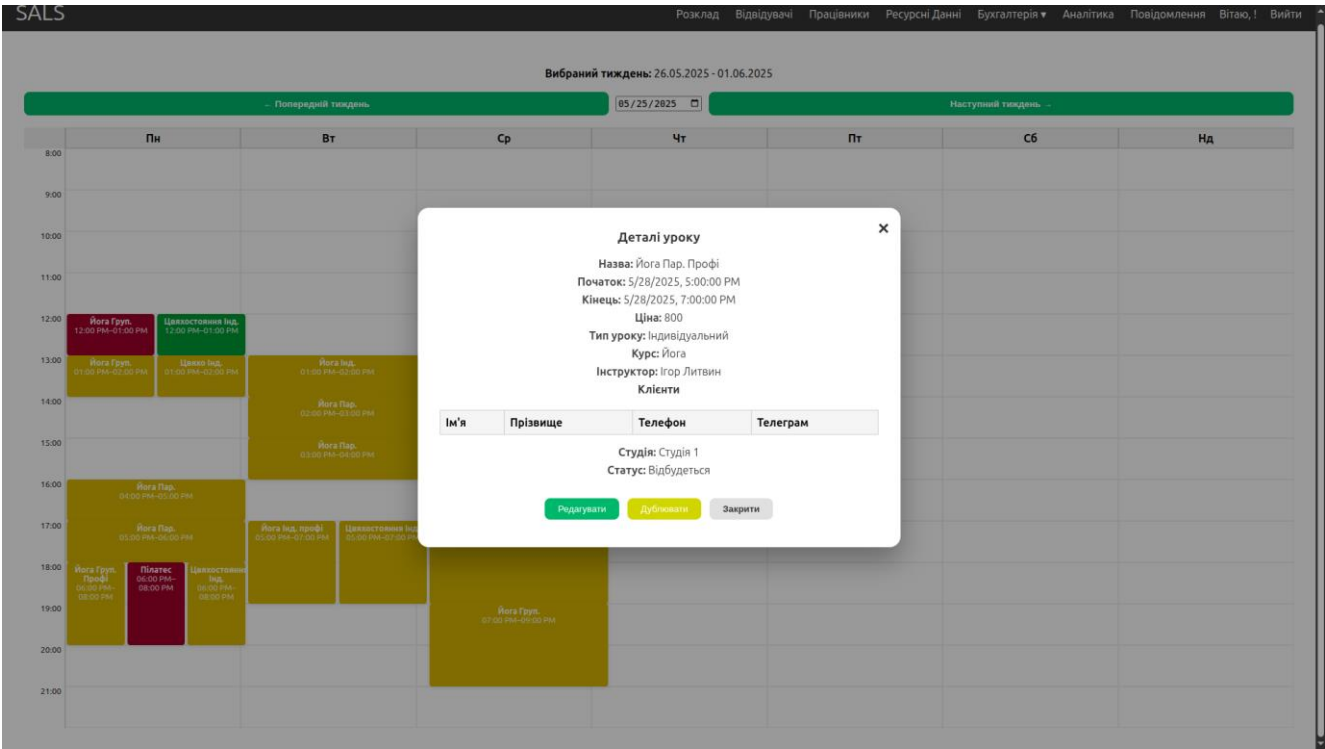


Рисунок 3.2 – Деталі уроку

За необхідності користувач може редагувати властивості уроку або додавати відвідувачів у модальному вікні (рис. 3.3).

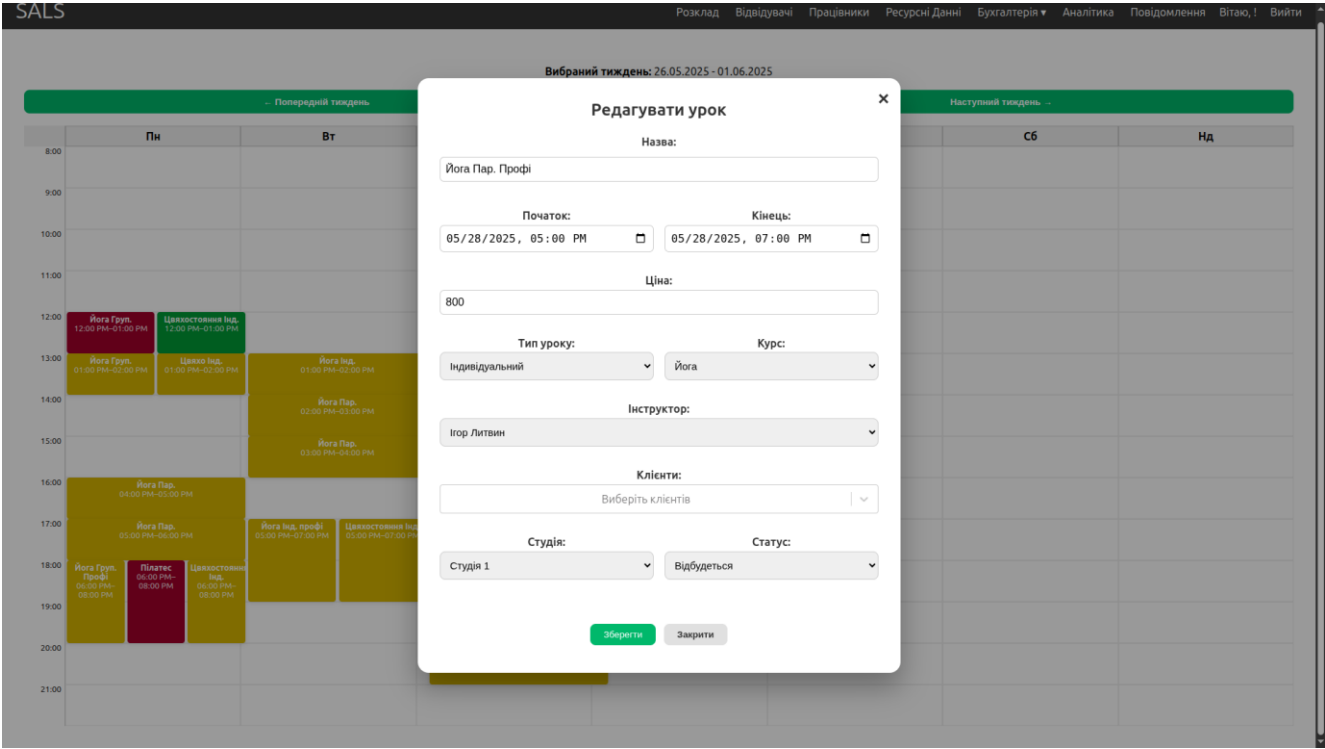


Рисунок 3.3 – Редагування уроку

Також існує можливість дублювати існуючий урок (рис 3.4).

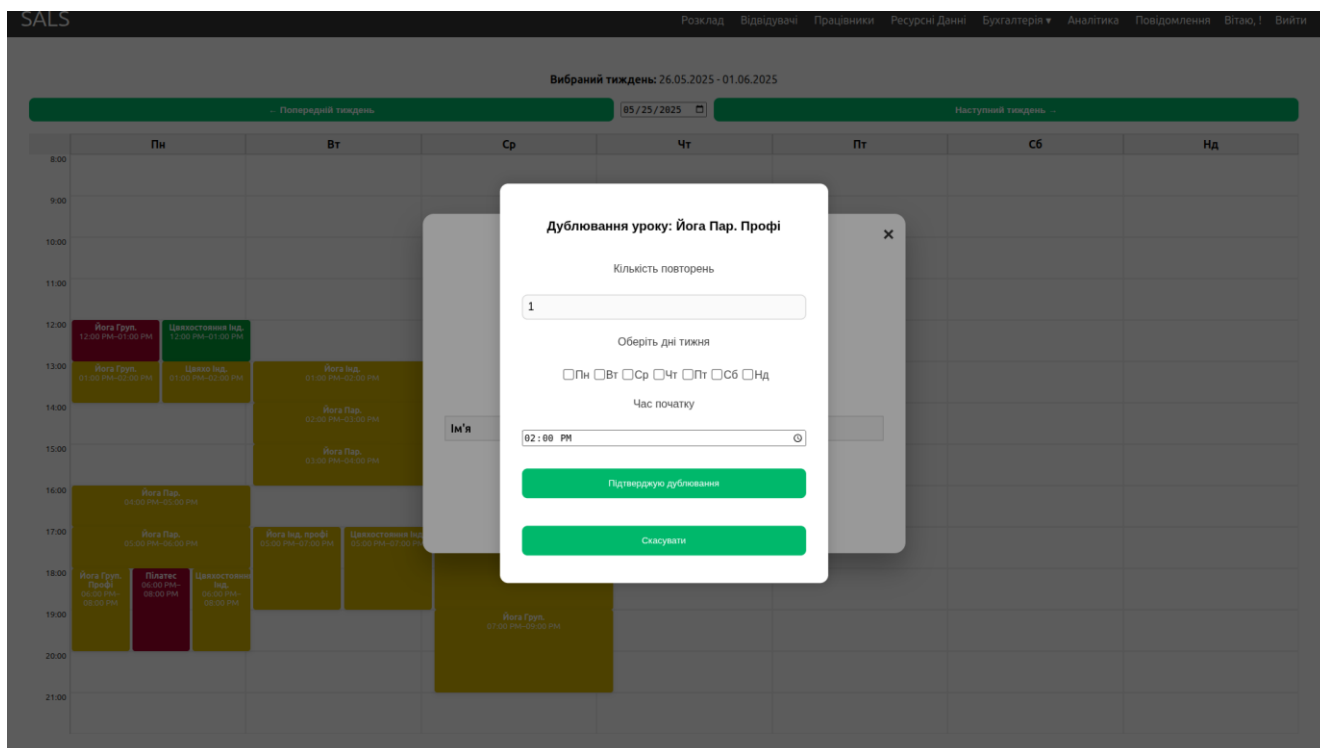


Рисунок 3.4 – Дублювання уроку

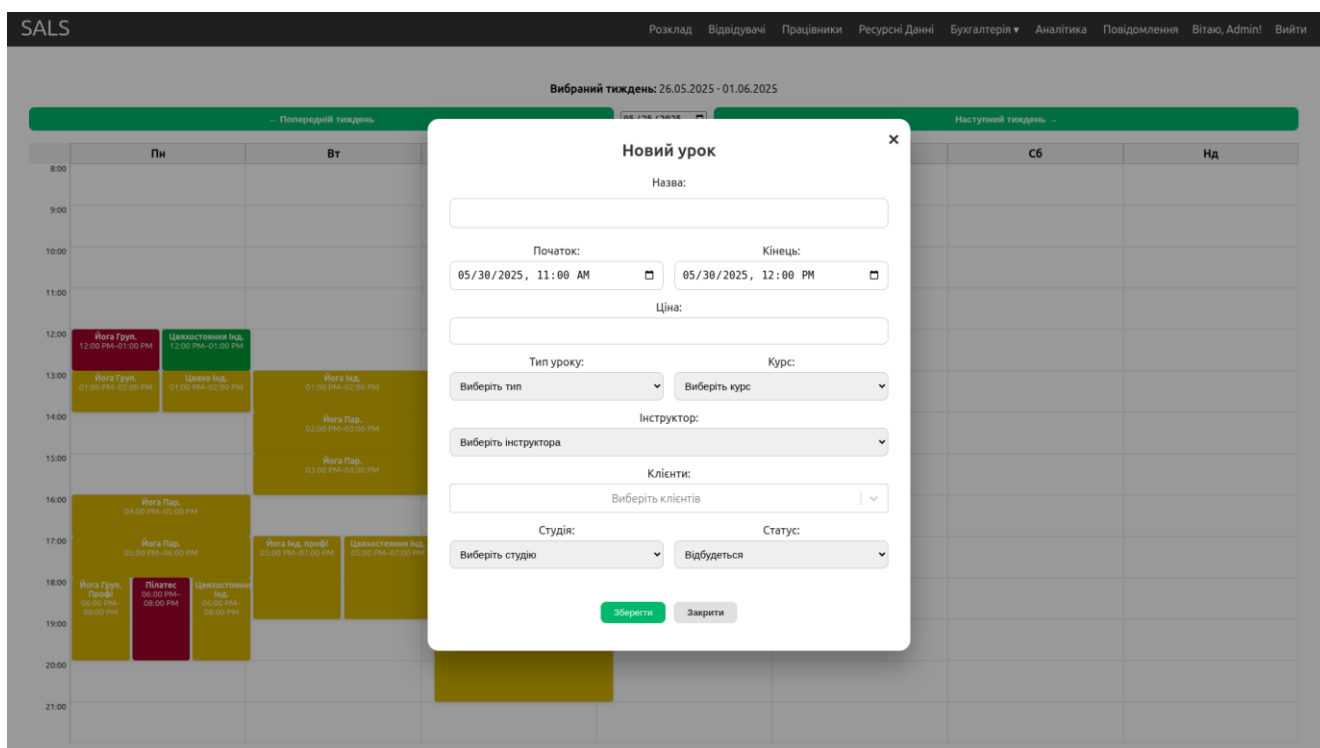


Рисунок 3.5 – Створення нового уроку

Адміністратор може створити новий урок у модальному вікні (рис. 3.5) після натискання на бажану клітинку розкладу.

Для роботи з даними відвідувача адміністратор може перейти на вкладку «Відвідувачі». Де він бачитиме всіх відвідувачів (рис. 3.6). Настистувши на кнопку

редагувати біля конкретного відвідувача з’являється модальне вікно з розширеними даними цього користувача та його абонементом. Адміністратор може редагувати дані користувача або додати абонемент (рис 3.7).

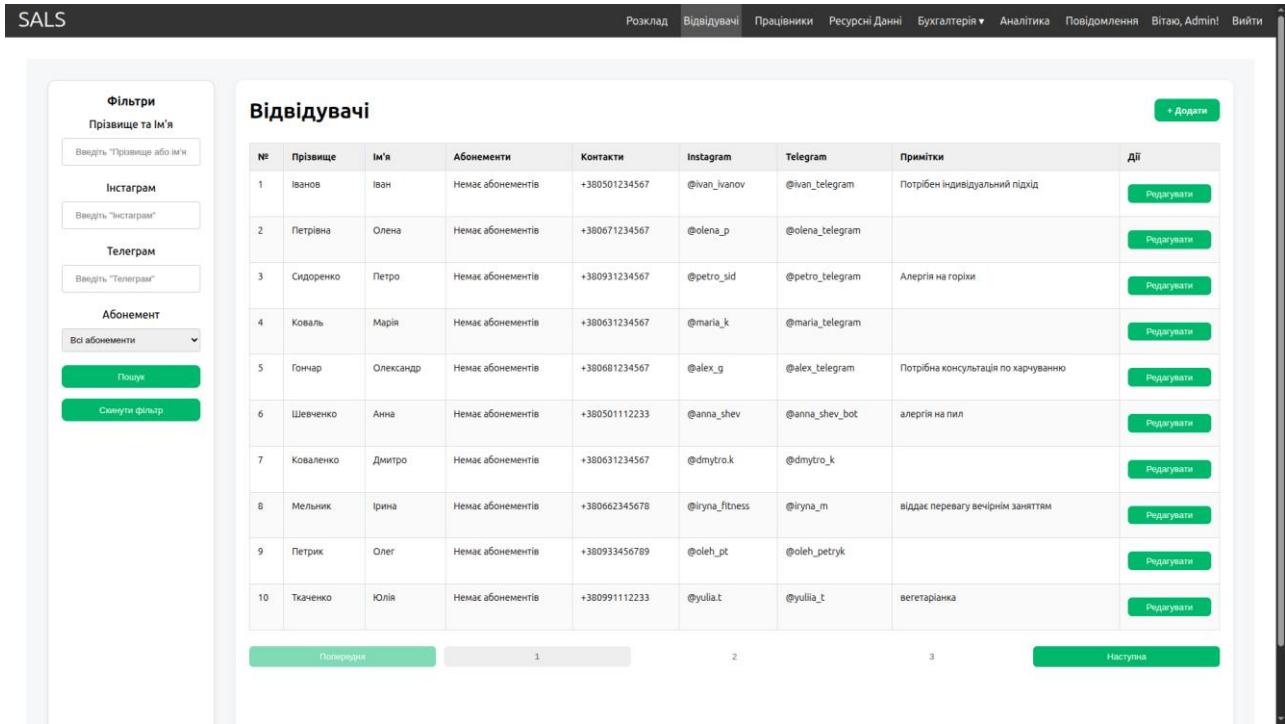


Рисунок 3.6 – Вкладка «відвідувачі»

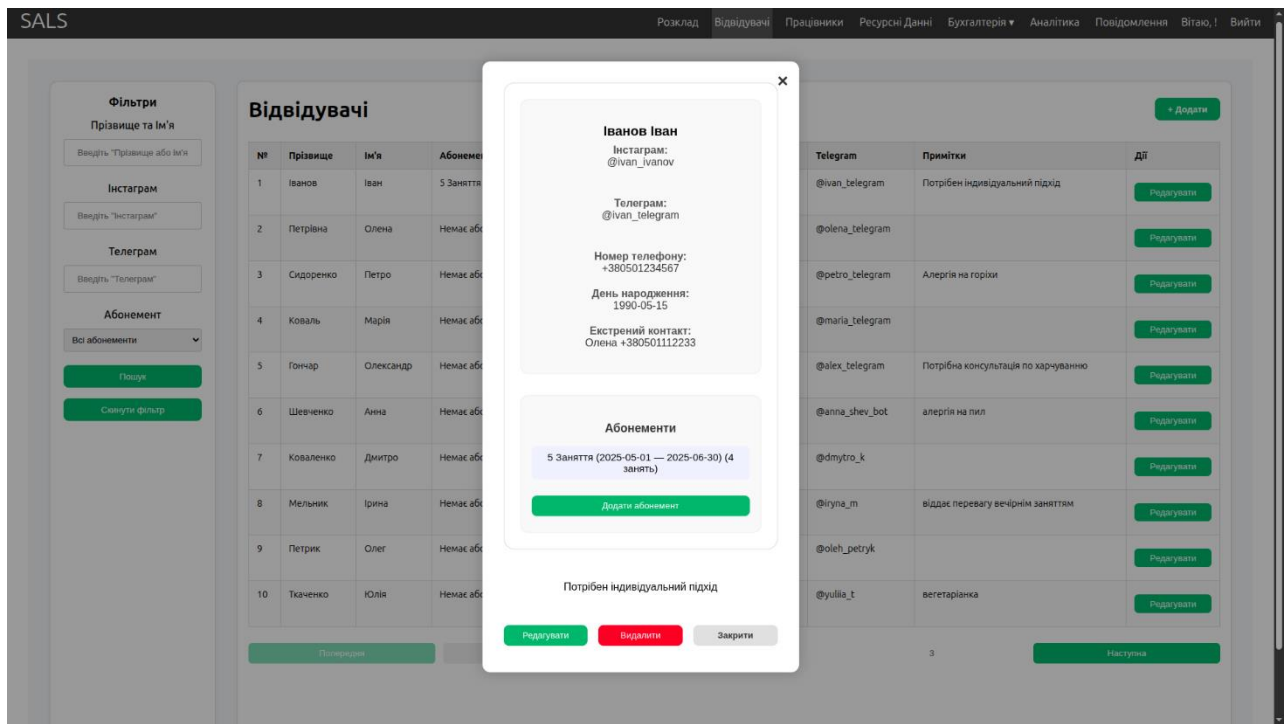


Рисунок 3.7 – Модальне вікно відвідувача

Для швидкого пошуку необхідного відвідувача є поле «фільтри» (рис. 3.8). Також адміністратор може додати нового відвідувача у систему (рис. 3.9).

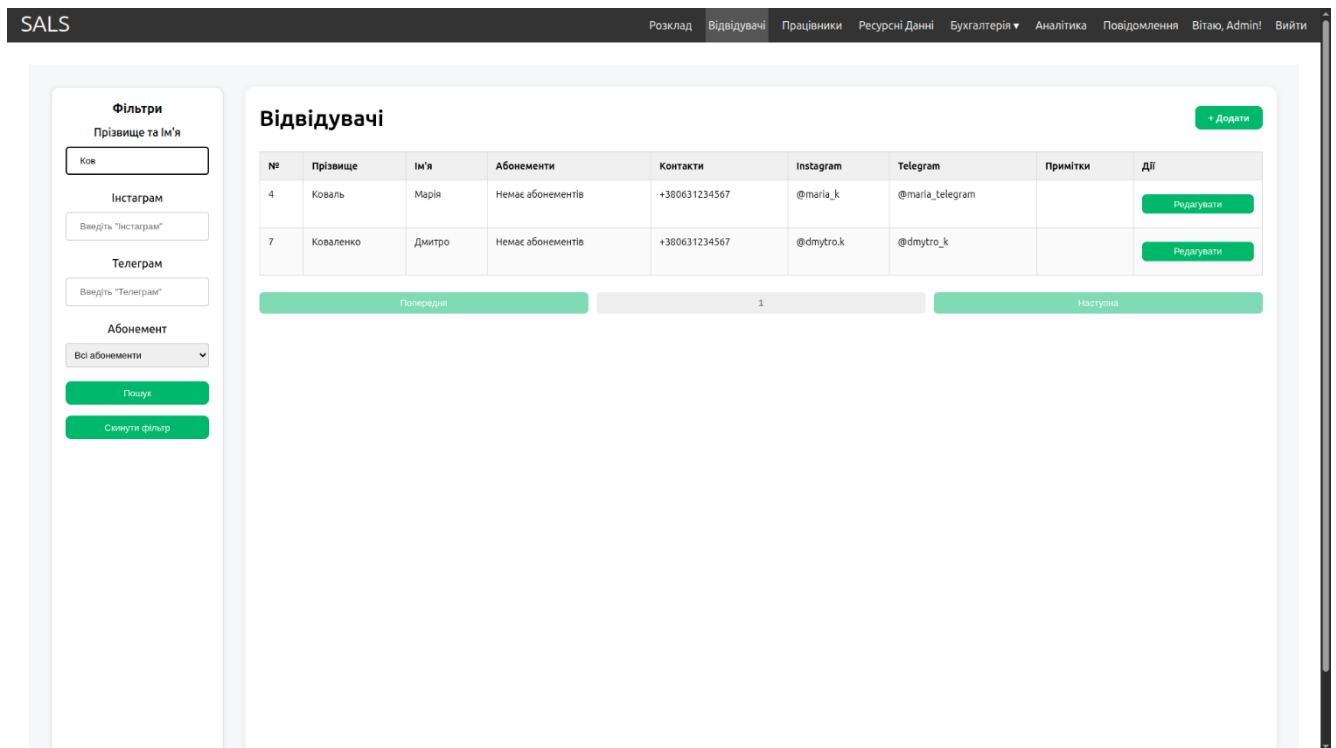


Рисунок 3.8 – Пошук відвідувача за прізвищем

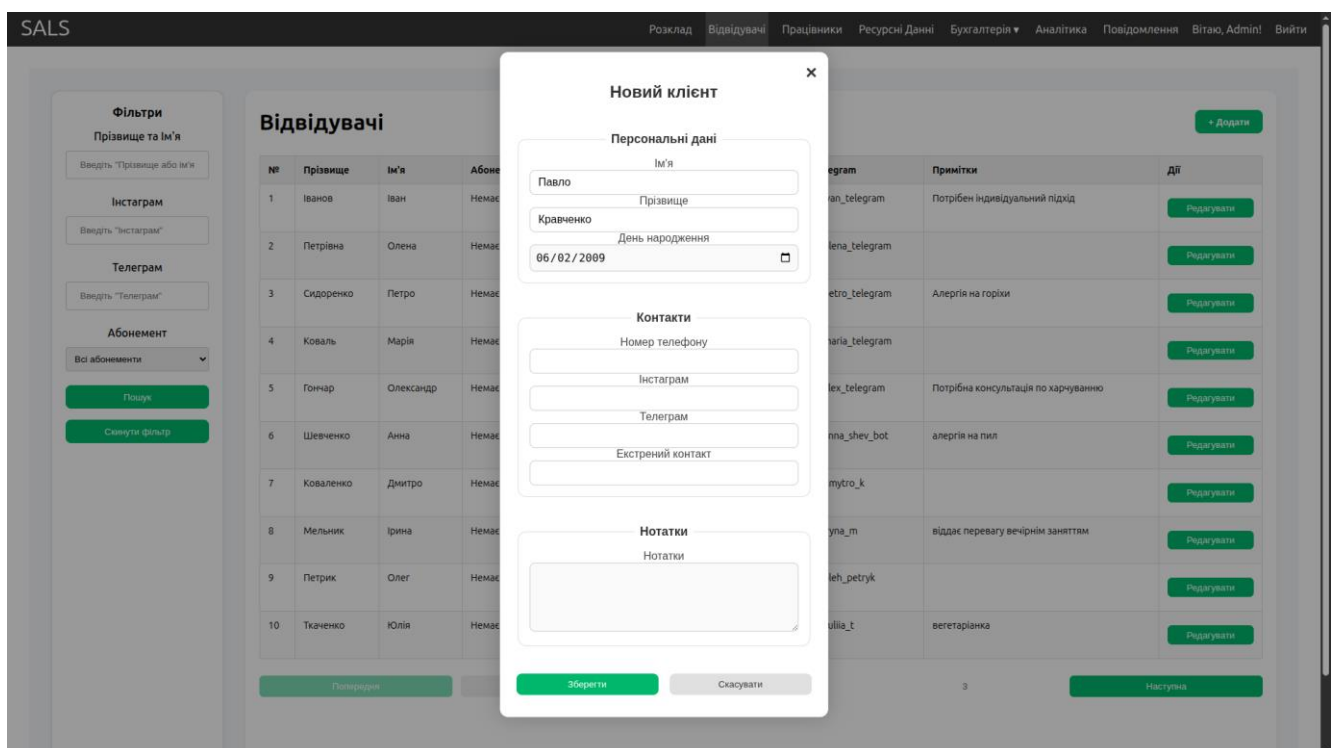


Рисунок 3.9 – Вікно додавання нового відвідувача

За зразком вкладки «Відвідувачі» є вкладка «Працівники», де адміністратор може переглянути всіх працівників (рис. 3.10), або відфільтрувати їх за певним значенням (рис. 3.11). Адміністратор може створити, редагувати, видалити уже існуючого працівника або просто переглянути дані про нього (рис 3.12).

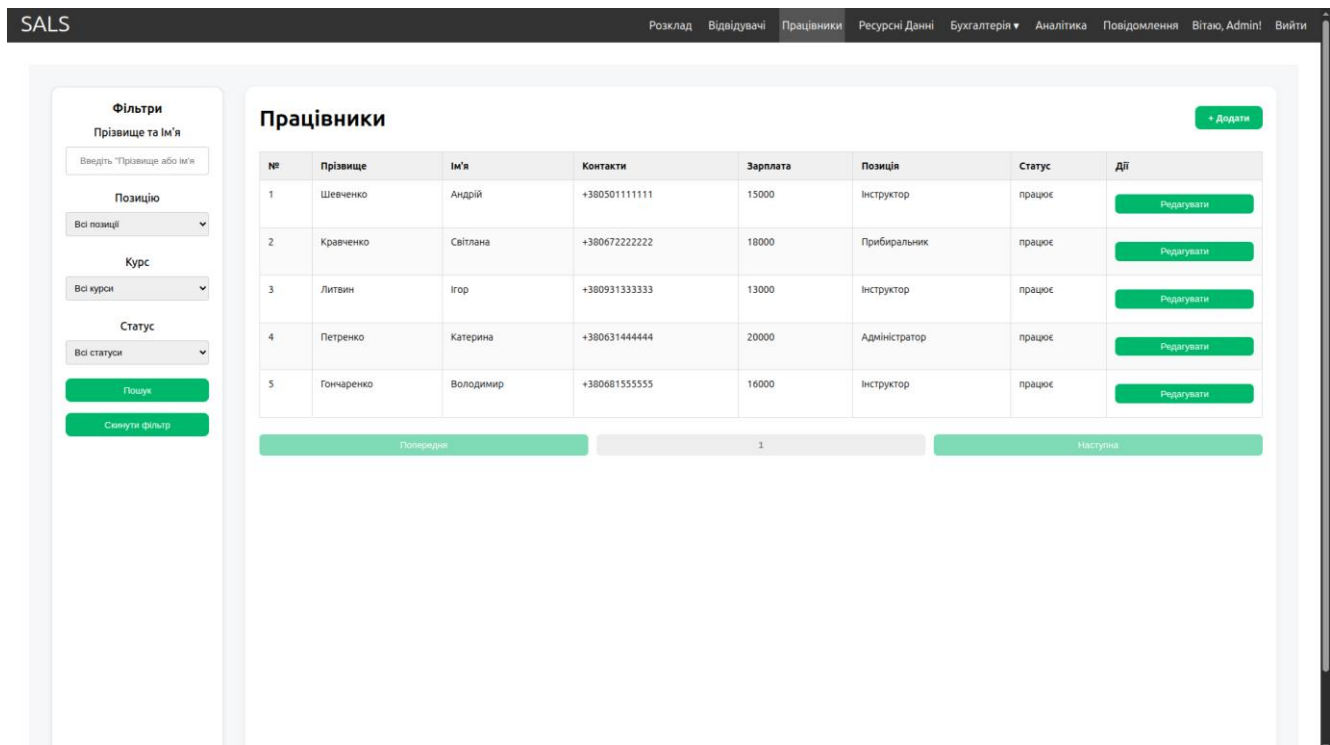


Рисунок 3.10 – Вкладка «Працівники»

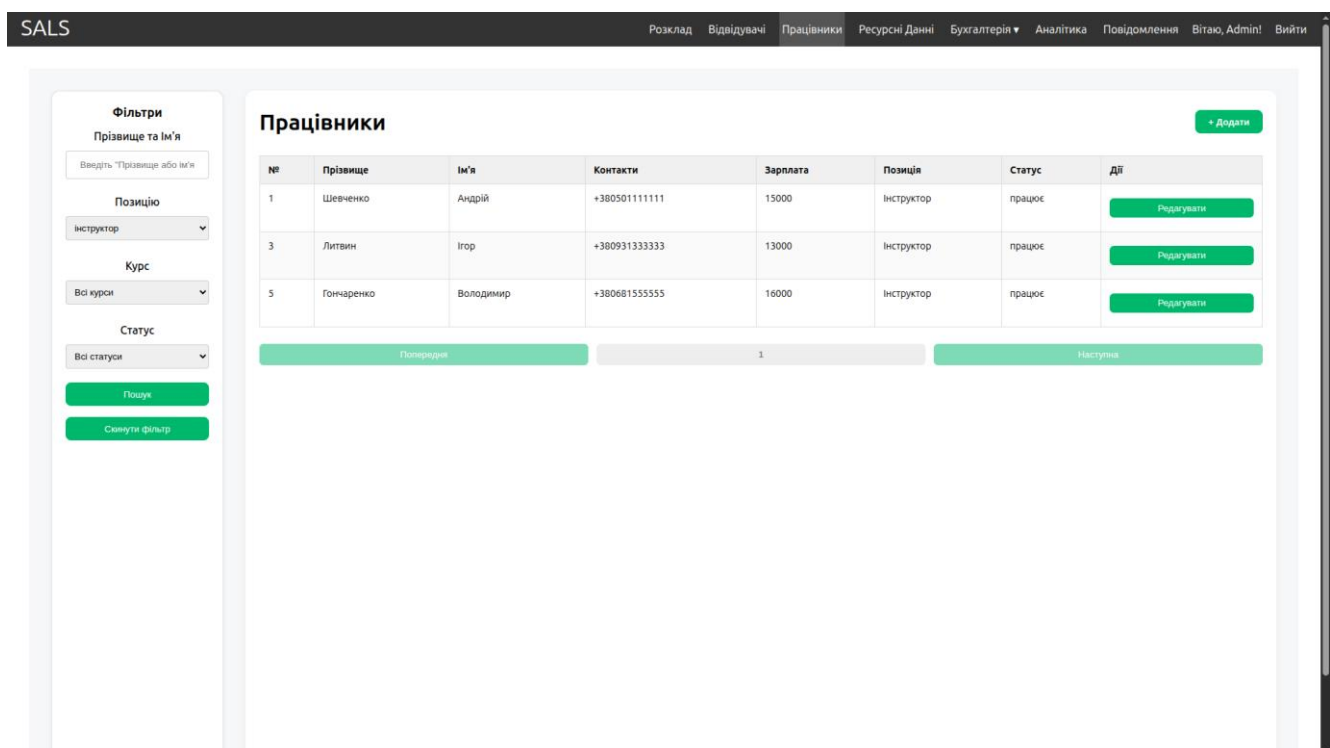


Рисунок 3.11 – Фільтр працівників за позицією

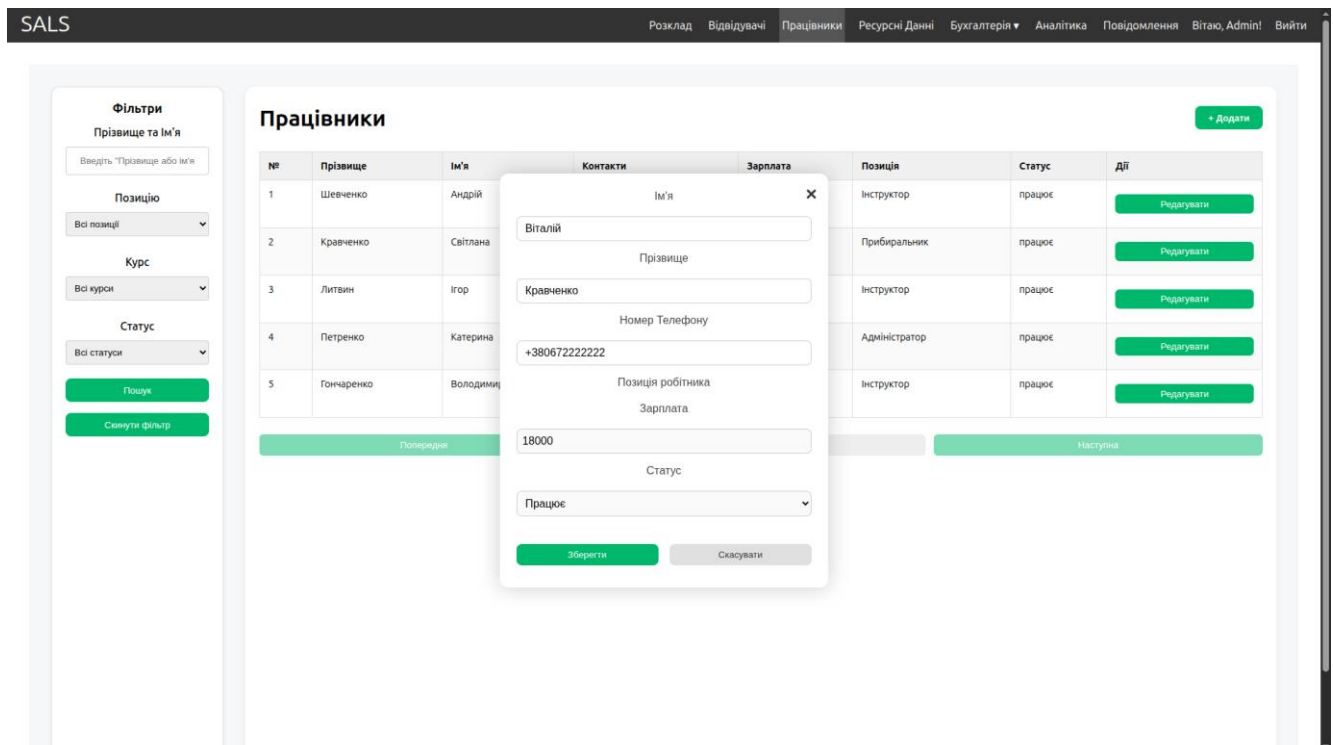


Рисунок 3.12 – Редагування працівника

Модуль «Бухгалтерія» який відповідає за фінанси системи складається з двох частин. Транзакції (рис. 3.13) – відображає усі прибутки, витрати та інші фінансові операції студії. Зарплати (рис. 3.14) - відображає автоматично підраховані заробітні плати співробітників студії.

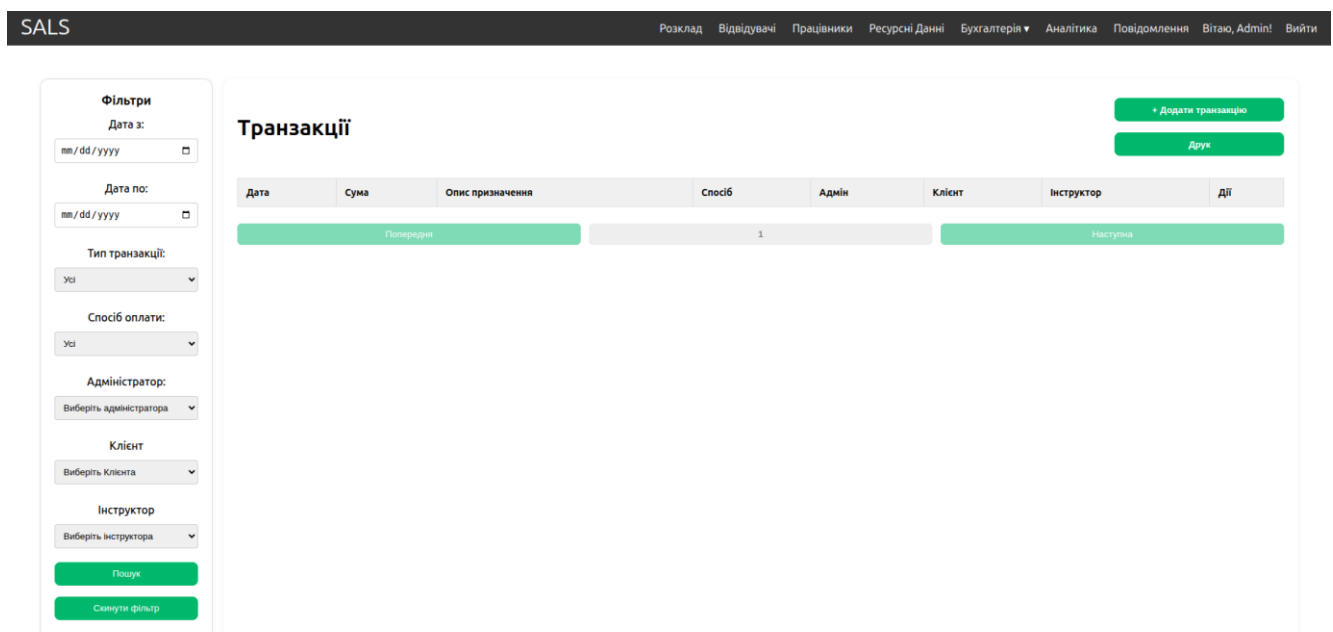


Рисунок 3.13 – Вкладка «Транзакції»

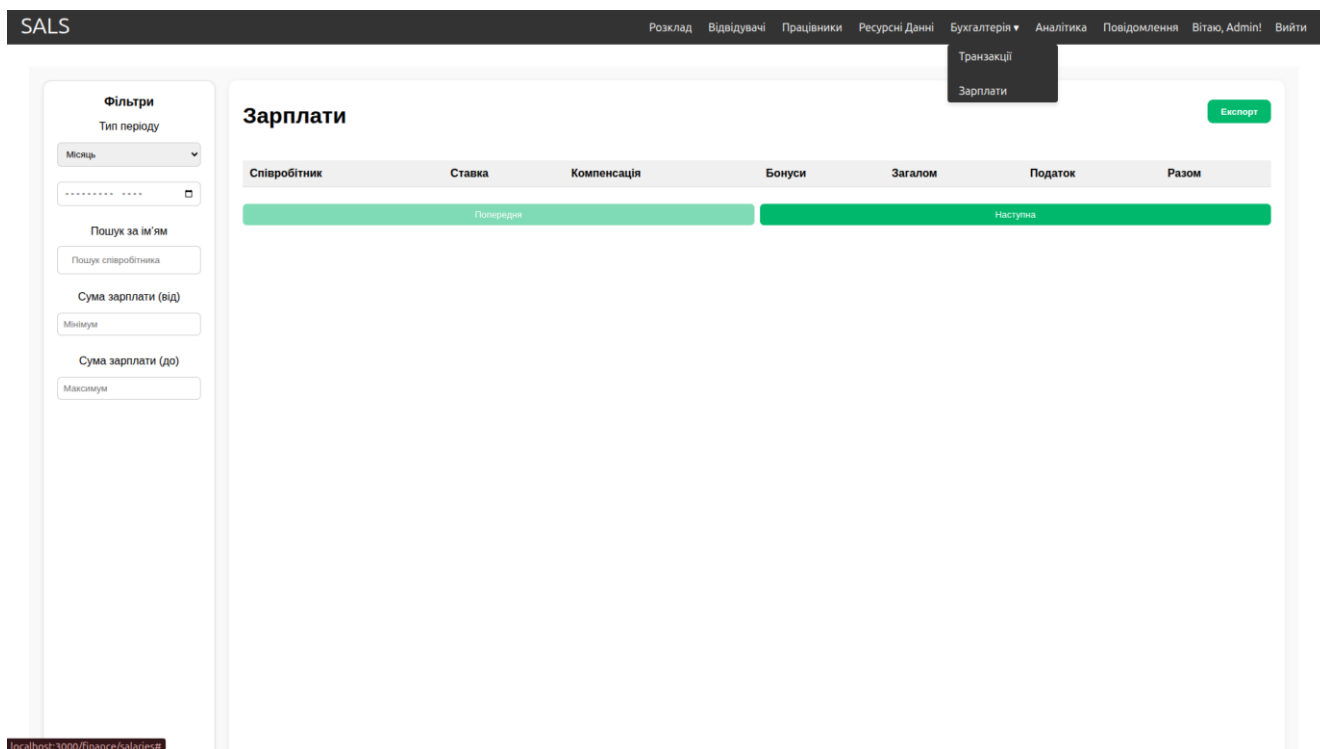


Рисунок 3.14 – Вкладка «Зарплати»

Для зручності співробітництва з бухгалтерами в системі реалізований експорт фінансових даних (рис. 3.15).

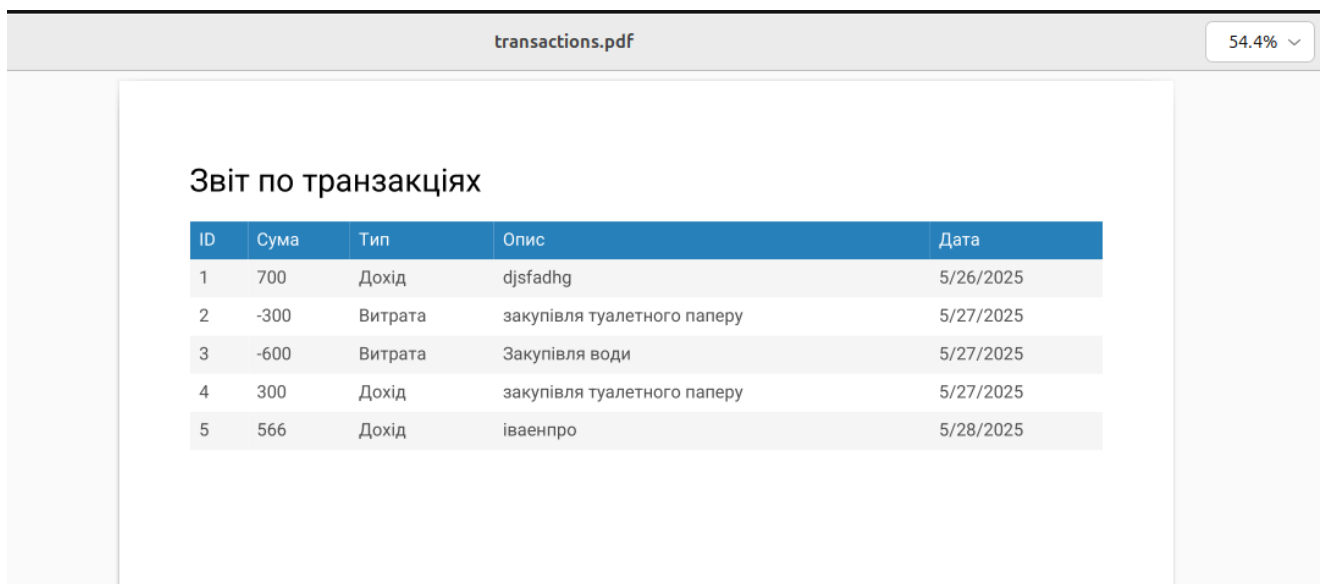


Рисунок 3.15 – Експортований файл з транзакціями

Система збирає аналітику за декількома параметрами. Адміністратор може переглянути які курси є популярнішими за інші (рис. 3.16), який інструктор є ефективнішим (рис. 3.17), проаналізувати фінанси або відвідуваність.

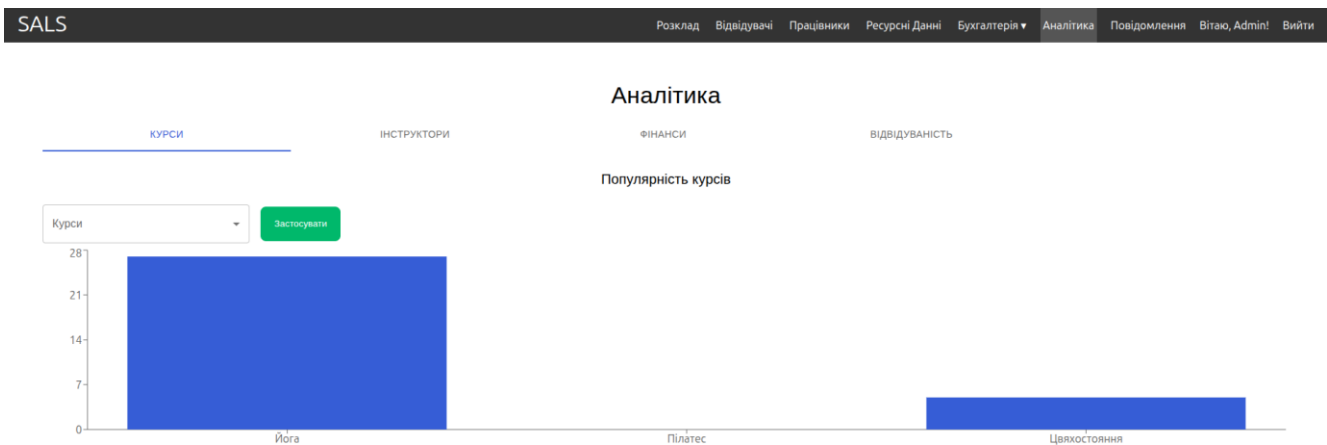


Рисунок 3.16 – Аналітика по курсам

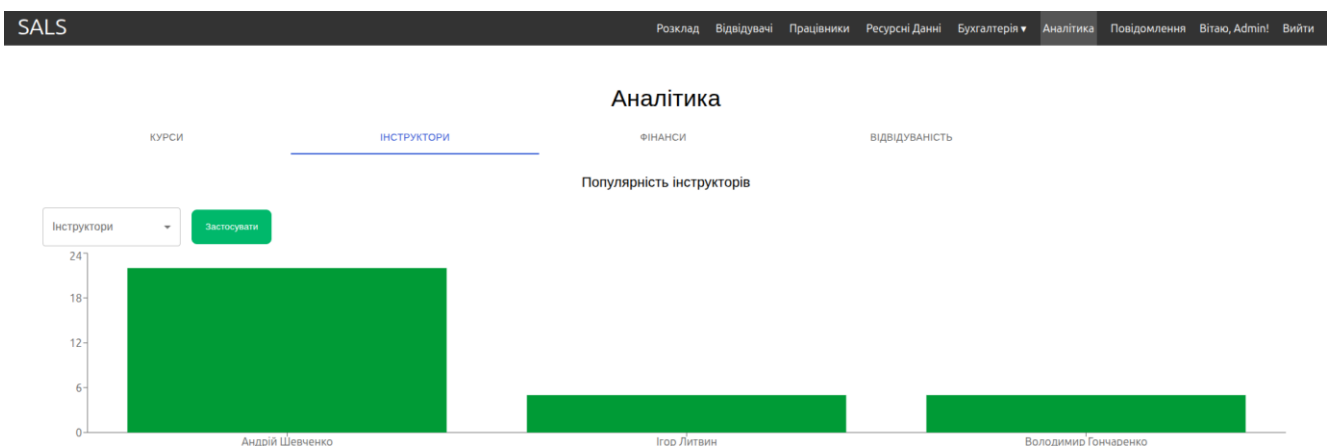


Рисунок 3.17 – Аналітика по інструкторам

Модуль «Повідомлення» (рис. 3.18) відображає повідомлення, що приходять від відвідувачів студії через Telegram-бот. Він забирає необхідність постійно перевіряти соціальні мережі на наявність нових повідомлень.

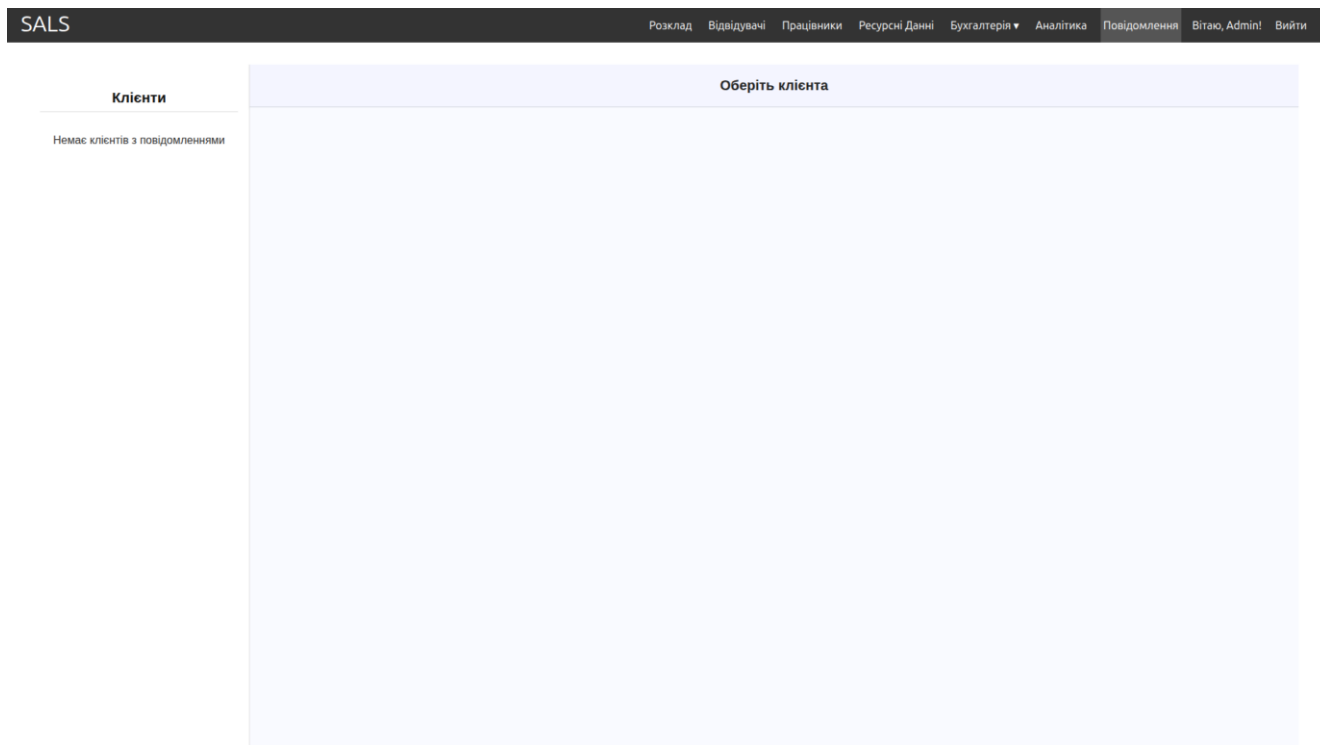


Рисунок 3.18 – Модуль «повідомлення»

Для керування дрібними даними до прикладу «типи занять» є модуль «ресурсні дані», де адміністратор може легко щось додати чи змінити (рис. 3.18 – рис. 3.23).

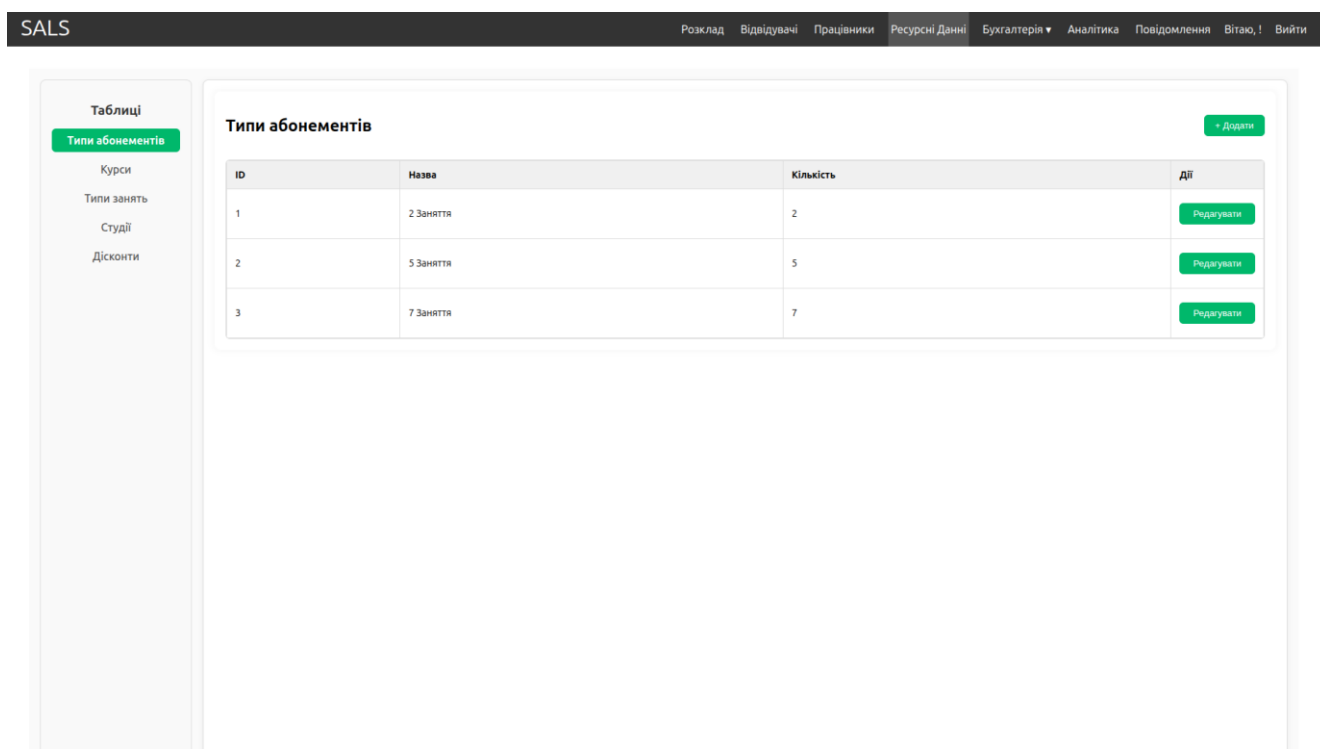


Рисунок 3.19 – Вкладка «Типи абонементів»

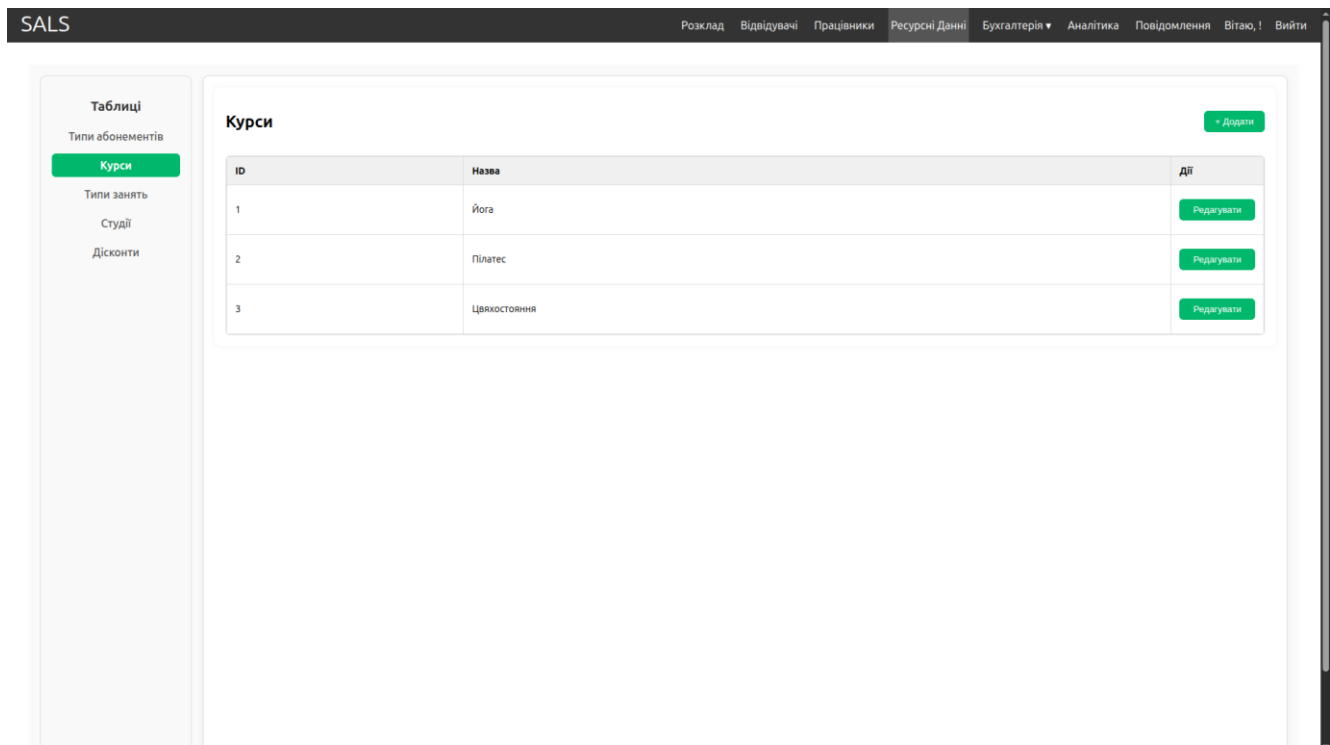


Рисунок 3.20 – Вкладка «Курси»

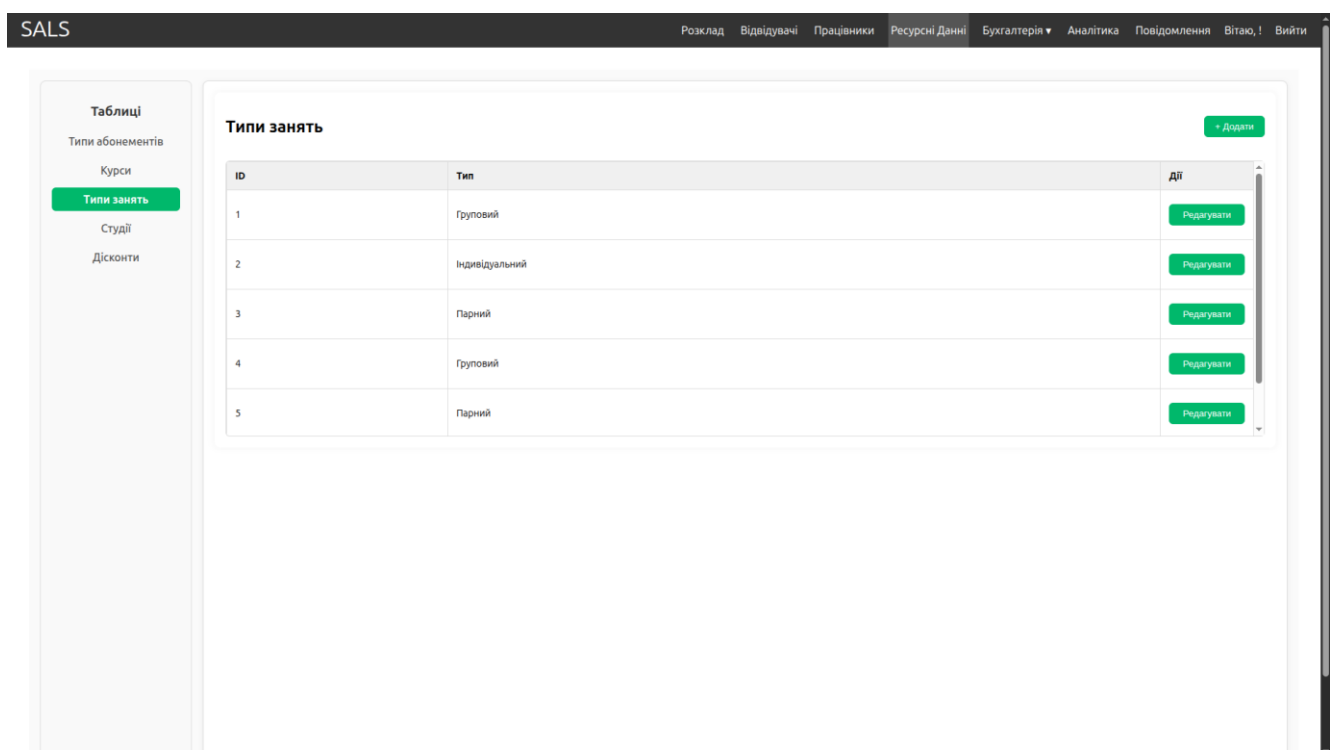


Рисунок 3.21 – Вкладка «Типи занять»

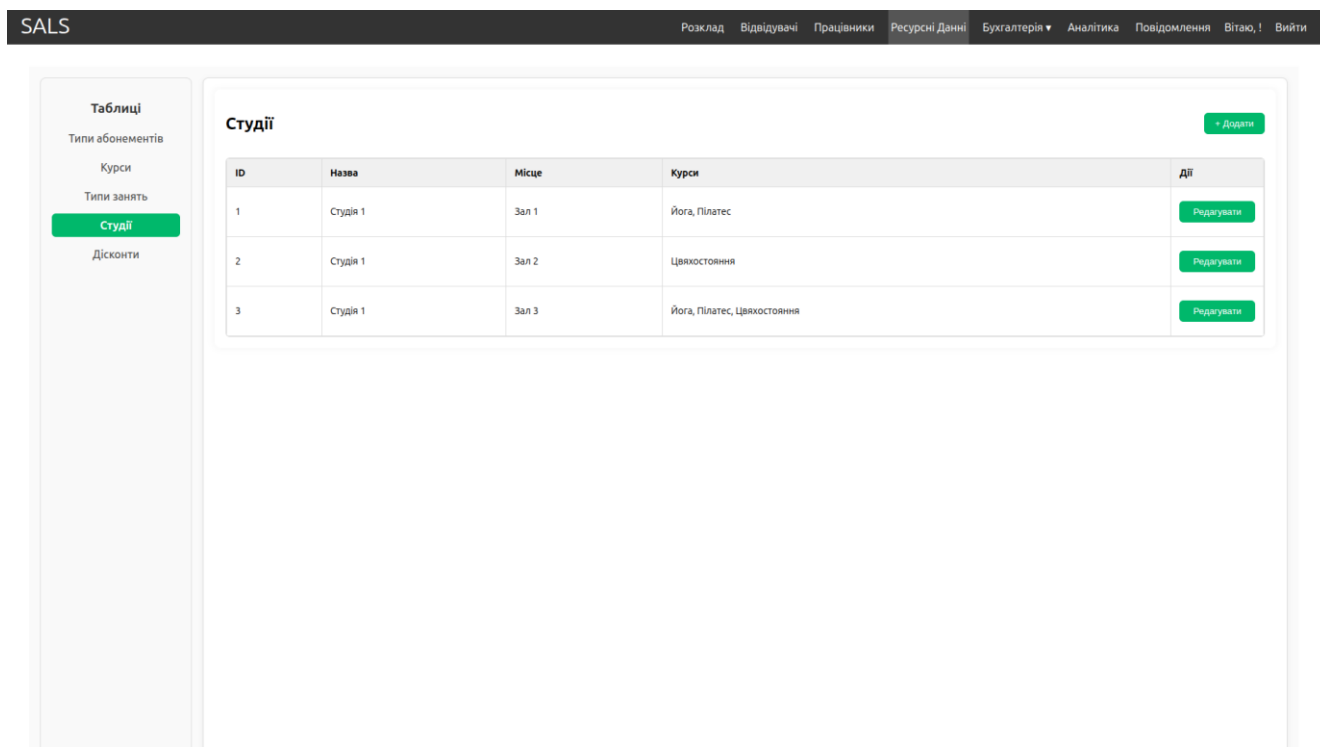


Рисунок 3.22 – Вкладка «Студії»

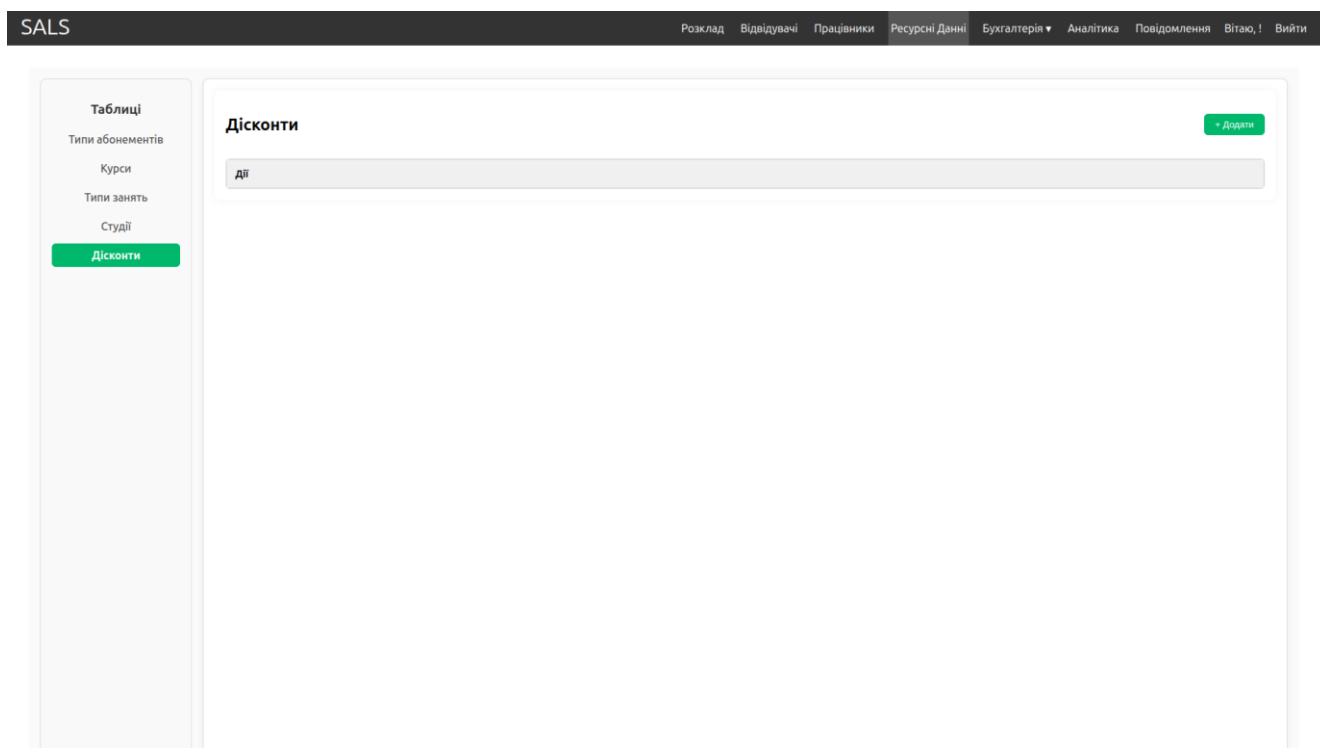


Рисунок 3.23 – Вкладка «Діскинти»

Інтерфейс користувача для відвідувача реалізовано за допомогою Telegram-бота, що забезпечує швидкий та зручний доступ до системи без необхідності встановлювати додаткові програми. Завдяки широкій популярності месенджера

Telegram та його підтримці на більшості мобільних пристроїв, користувачі можуть взаємодіяти з ботом у звичному для себе середовищі, що значно підвищує доступність і комфорт використання. Для початку роботи з ботом необхідно просто зайти у нього (рис. 3.24).

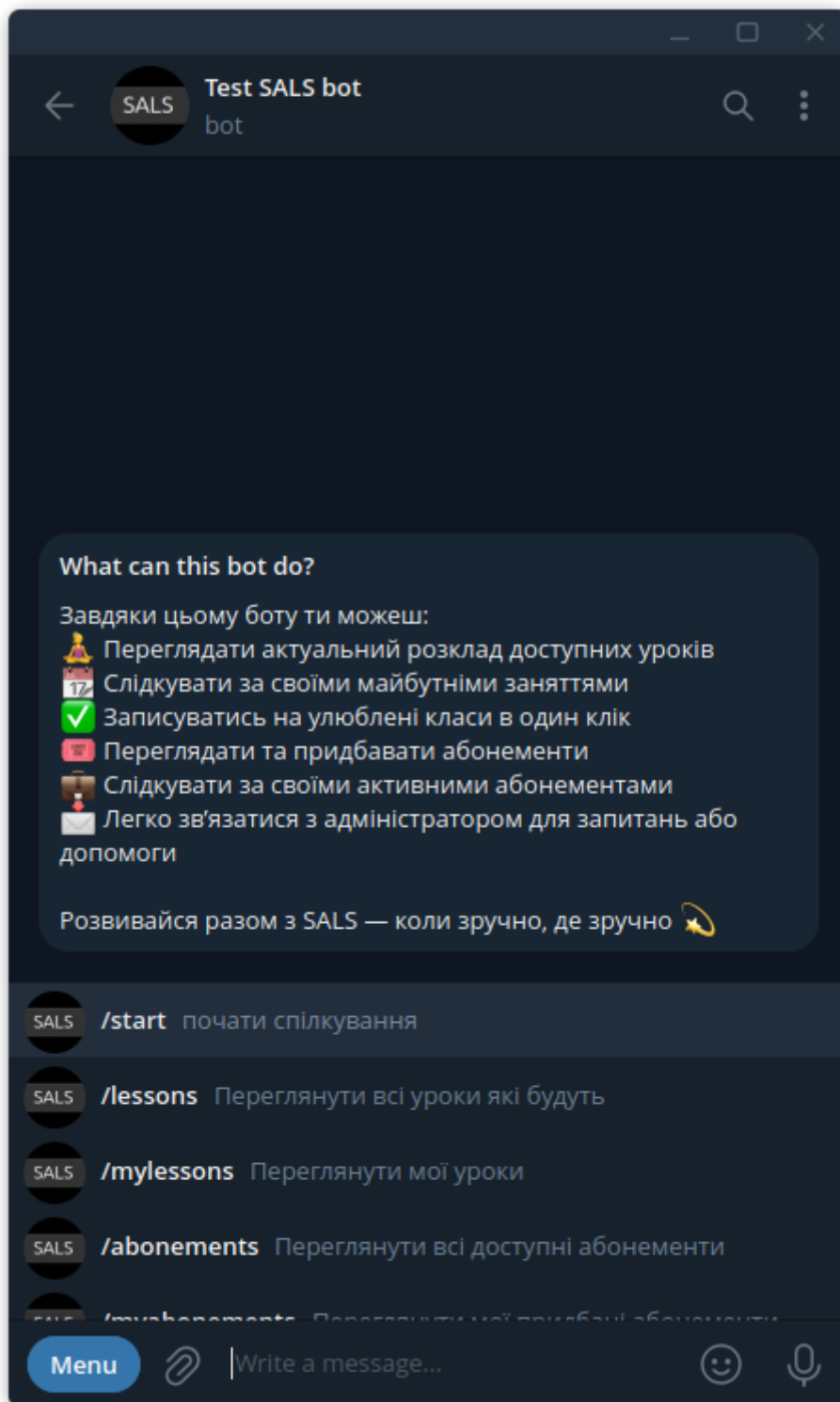


Рисунок 3.24 – Початок роботи з ботом

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		58

Програма запропонує користувачу зареєструватися у системі (рис. 3.25), надсилаючи відповідні повідомлення.

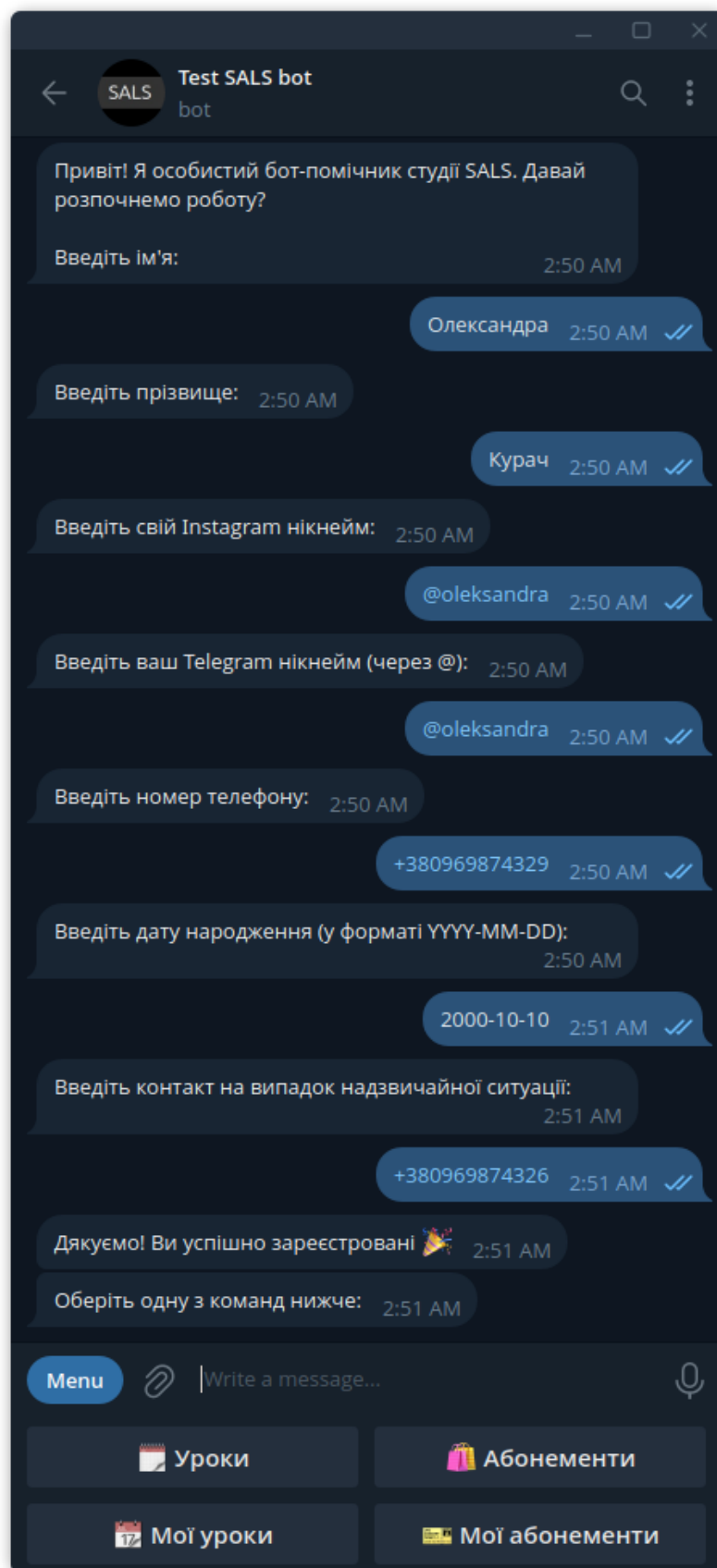


Рисунок 3.25 – Реєстрація у боті

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
						59
Змн.	Арк.	№ докум.	Підпис	Дата		

Інтерфейс Telegram-бота розроблений таким чином, щоб бути максимально простим, інтуїтивно зрозумілим і дружнім до користувача. Використання кнопок, меню та швидких відповідей забезпечує легку навігацію, навіть для користувачів без технічного досвіду. Взаємодія з ботом не потребує введення складних команд — усі дії здійснюються через зрозумілі елементи управління.

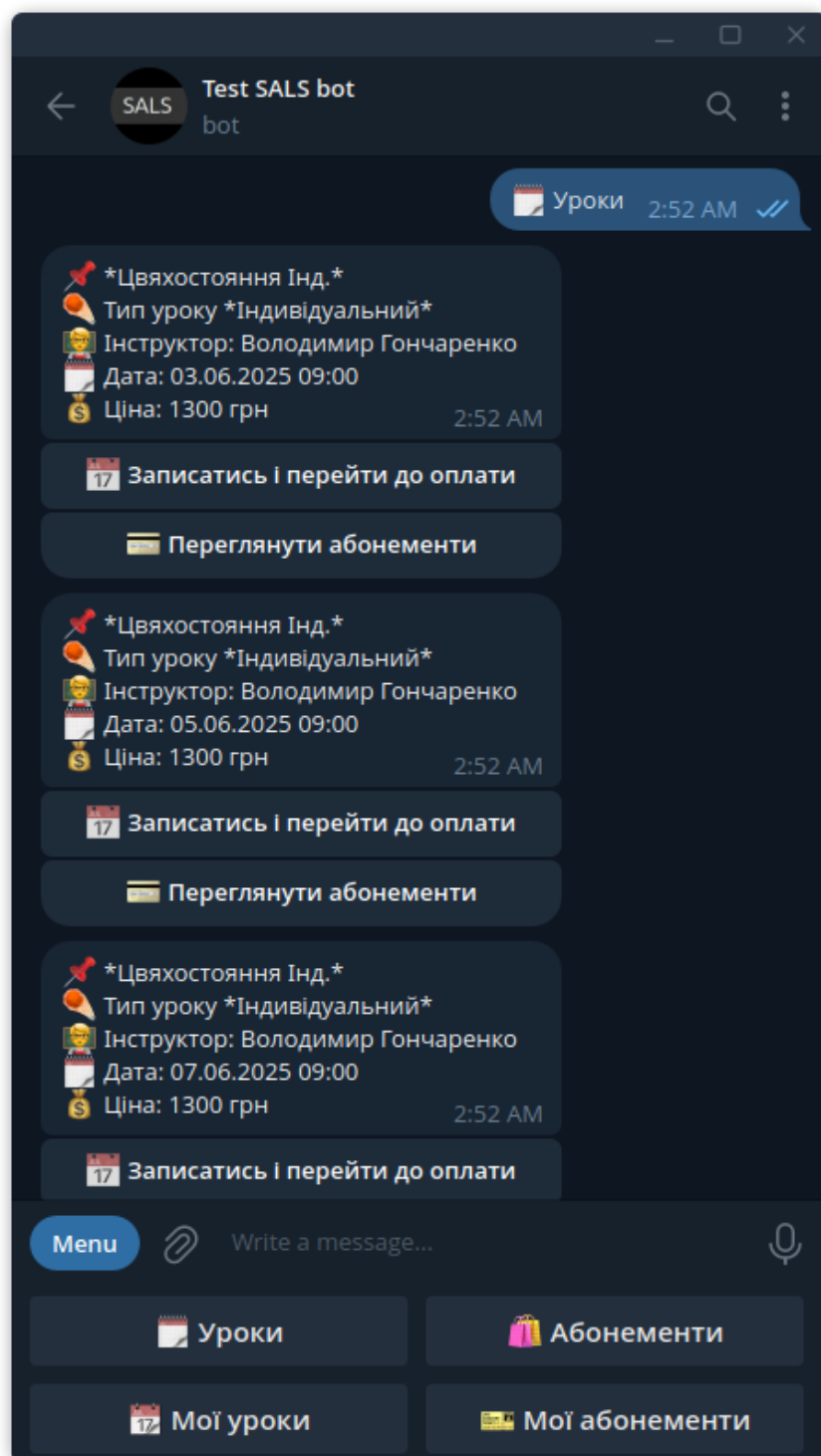


Рисунок 3.26 – Перегляд уроків

Запис на уроки зроблений через зручну систему вибору часу та типу заняття дозволяє швидко зареєструватися на обраний урок (рис. 3.27).

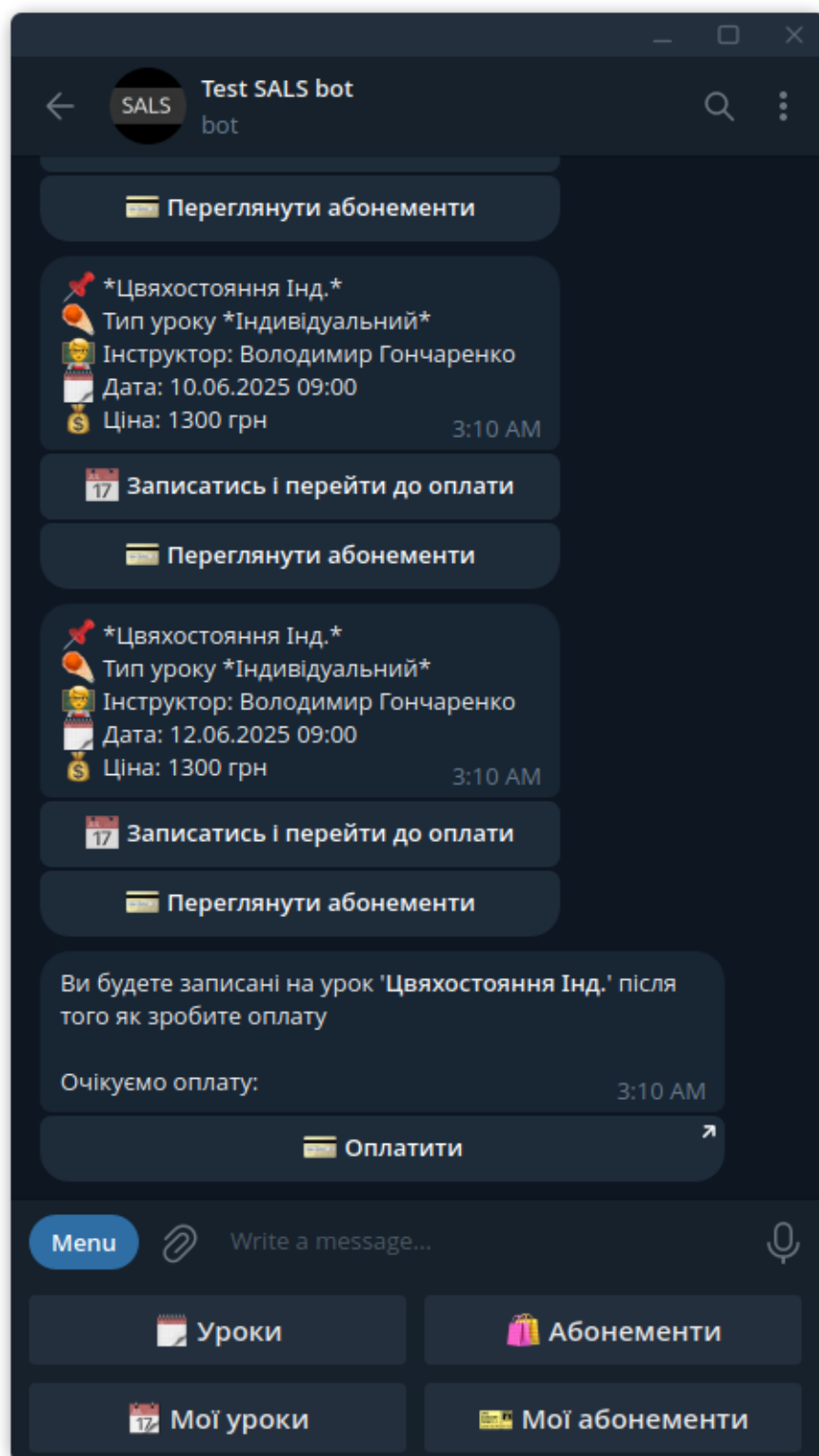


Рисунок 3.27 – Запис на заняття

Оплату обраного уроку можна здійснити безпосередньо через Telegram-бота за допомогою платіжної системи LiqPay (рис. 2.28), що надає зручний і безпечний

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
						61
Змн.	Арк.	№ докум.	Підпис	Дата		

спосіб проведення онлайн-платежів [30]. Після вибору уроку бот автоматично генерує посилання на оплату, яке відкривається у веб-браузері користувача. Сервіс підтримує оплату банківськими картками (Visa, Mastercard), Google Pay, Apple Pay та іншими популярними методами, що робить процес максимально швидким і комфортним.

QR-код для оплати

Використовуйте Privat24

Тестовий режим

LIQPAY >>

Дані про оплату

Оплата за урок: Цвяхостояння Інд.

До сплати: 1300.00 UAH

24 Pay

Google Pay

VISA

.... 5608

або

Картка

24

Приват24

Рахунок

Номер картки

4242 4242 4242 4242

VISA

Термін дії

CVV2

12/25

... 787

Ім'я власника картки*

Олександра

Прізвище власника картки*

Курач

☐ Відправити квитанцію на e-mail

Натискаючи на кнопку «Оплатити», ви підтверджуєте що ознайомлені з переліком інформації про послугу та приймаєте умови [публічного договору](#)

Оплатити 1300.00 UAH

Скасувати оплату

Рисунок 3.28 – Оплата

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
						62
Змн.	Арк.	№ докум.	Підпис	Дата		

Після успішної оплати бот надсилає підтвердження, а відповідна інформація зберігається в особистому кабінеті користувача та відображається в історії платежів. Це дозволяє забезпечити прозорість фінансових операцій і контроль за витратами (рис. 3.29 – рис. 3.30).

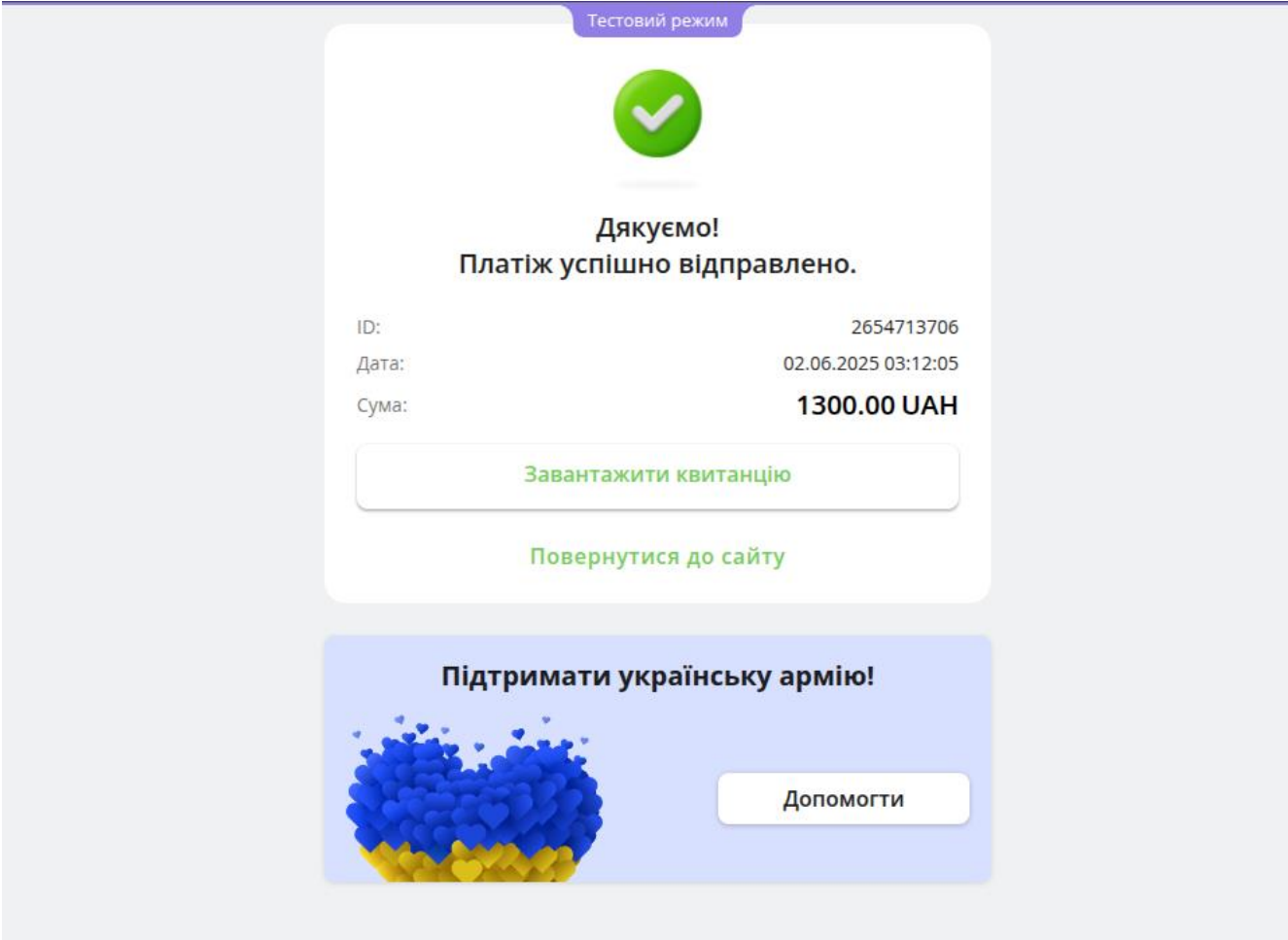


Рисунок 3.29 – Успішна оплата



Рисунок 3.30 – Повідомлення про успішну оплату

У разі змін планів користувач може легко скасувати участь у занятті або перенести запис на інший час (рис. 3.31 – рис. 3.32).

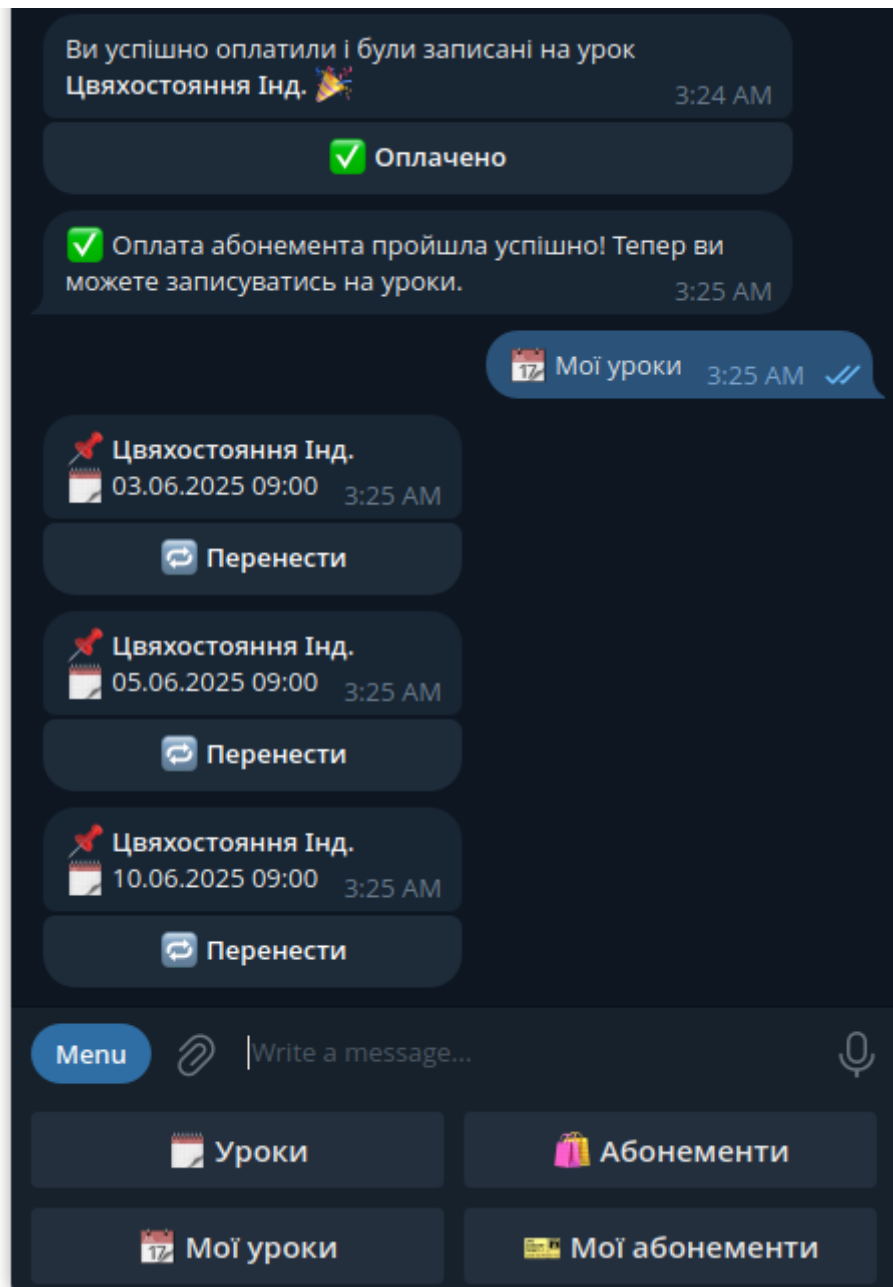


Рисунок 3.31 – Перегляд записів на урок

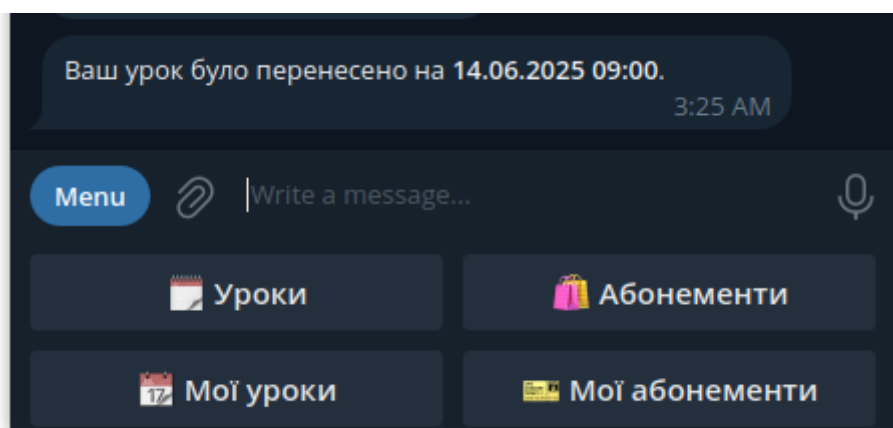


Рисунок 3.32 – Преренесення запису на урок

Крім звичайного запису на окремі заняття, користувач через Telegram-бота може переглянути всі доступні у студії абонементи. Це можуть бути, наприклад, пакети на 4, 8 або 12 занять з певною знижкою, або спеціальні пропозиції на конкретні курси чи часи (рис. 3.33).

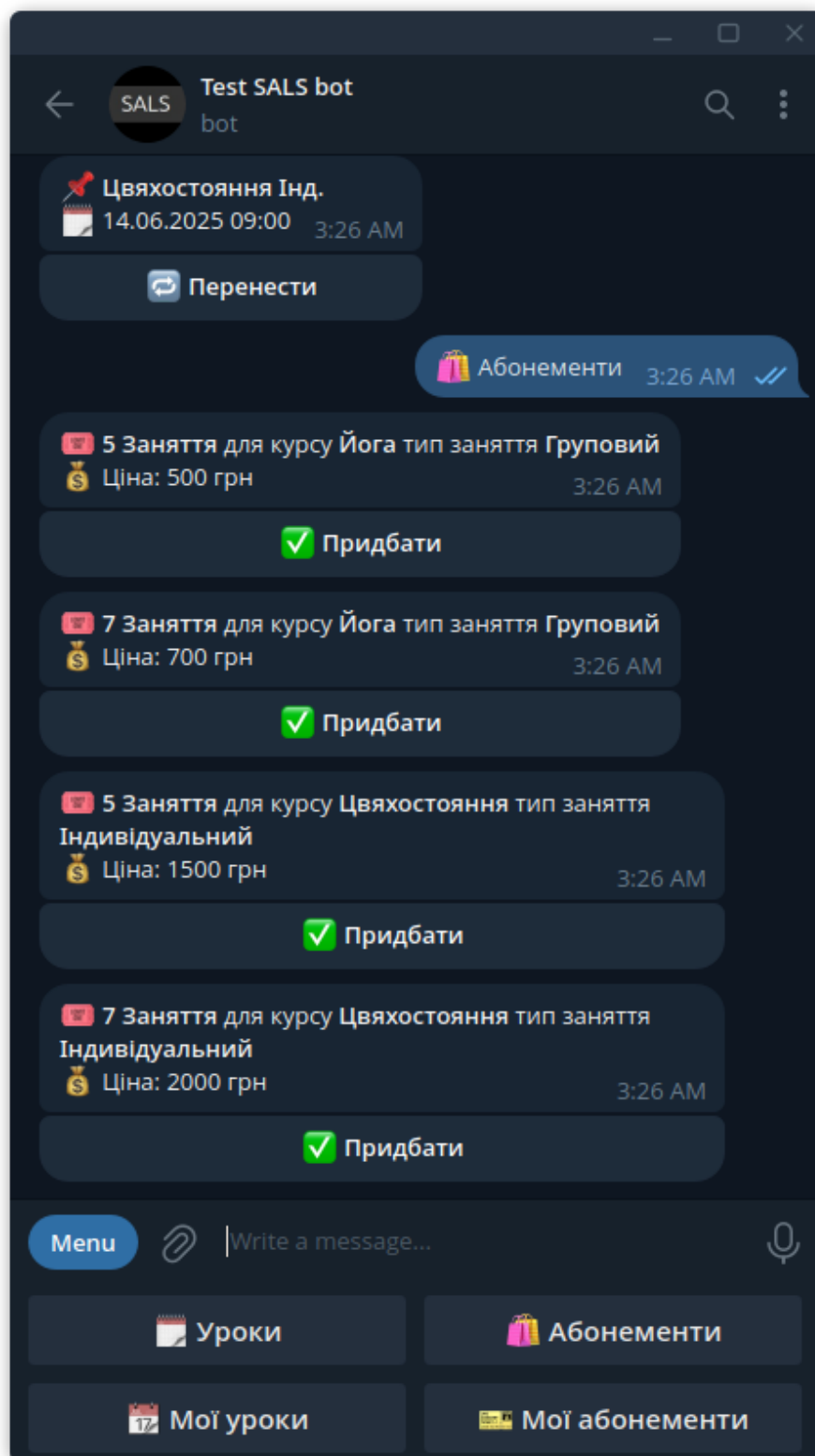


Рисунок 3.33 – Перегляд доступних абонементів

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		65

Якщо відвідувача зацікавив якийсь абонемент, він може одразу придбати його через бот. Для цього достатньо натиснути кнопку «Купити», після чого бот згенерує платіжне посилання через LiqPay. Після успішної оплати абонемент автоматично додається до профілю користувача. У будь-який момент він може відкрити список своїх абонементів, подивитися, скільки занять залишилося, коли вони дійсні до, і які умови діють (рис. 3.34 – рис. 3.35).

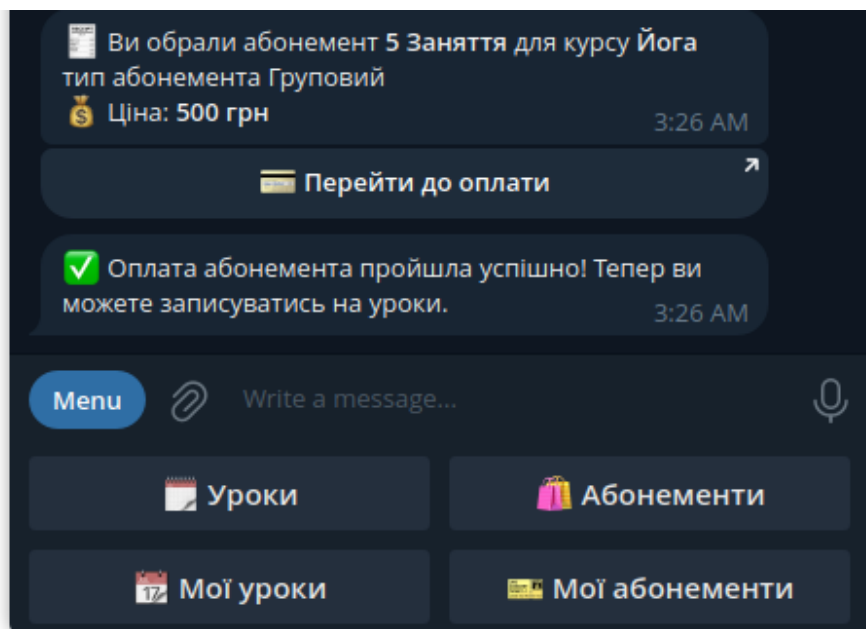


Рисунок 3.34 – Купівля абонементу

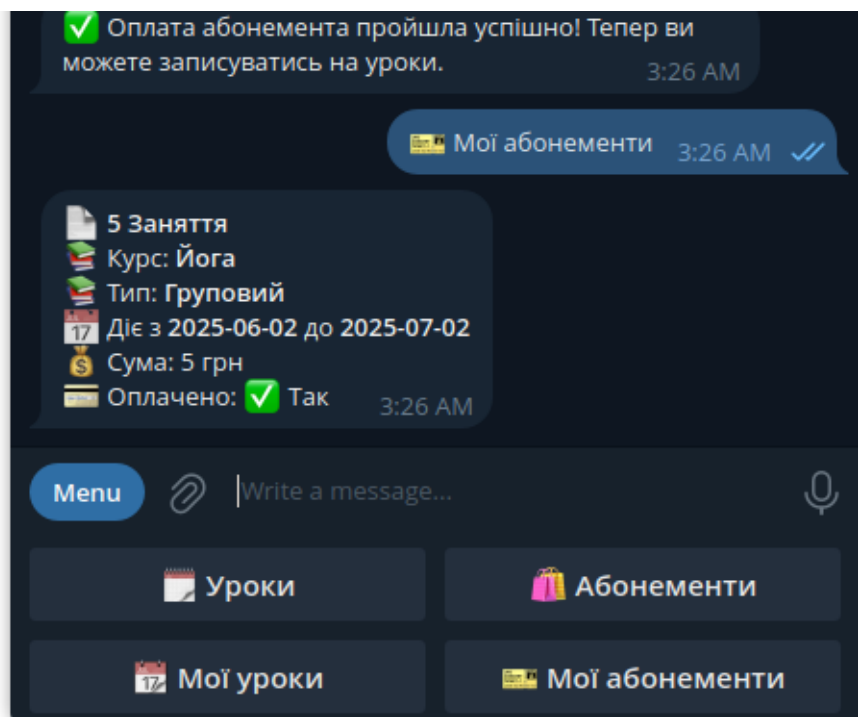


Рисунок 3.35 – Перегляд абонементів користувача

У разі виникнення запитань або будь-якої незрозумілої ситуації, відвідувач може безпосередньо написати адміністратору просто в Telegram-боті. Для цього в боті є окрема команда, де можна залишити повідомлення. Користувач просто вводить свій текст — і він автоматично надходить на адміністративну панель у розділ «Повідомлення». Це дозволяє швидко реагувати на звернення, не вимагає телефонних дзвінків або пошуку контактів. Адміністратор бачить повідомлення в режимі реального часу і може одразу відповісти або зв'язатися з користувачем для уточнення деталей. Такий зворотний зв'язок значно підвищує зручність і довіру до сервісу (рис. 3.36 – рисю 3.38).

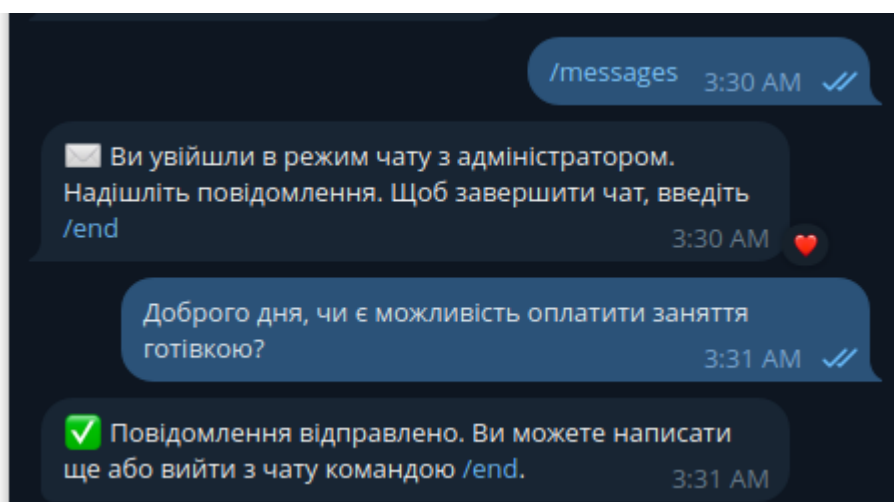


Рисунок 3.36 – Початок спілкування з адміністратором

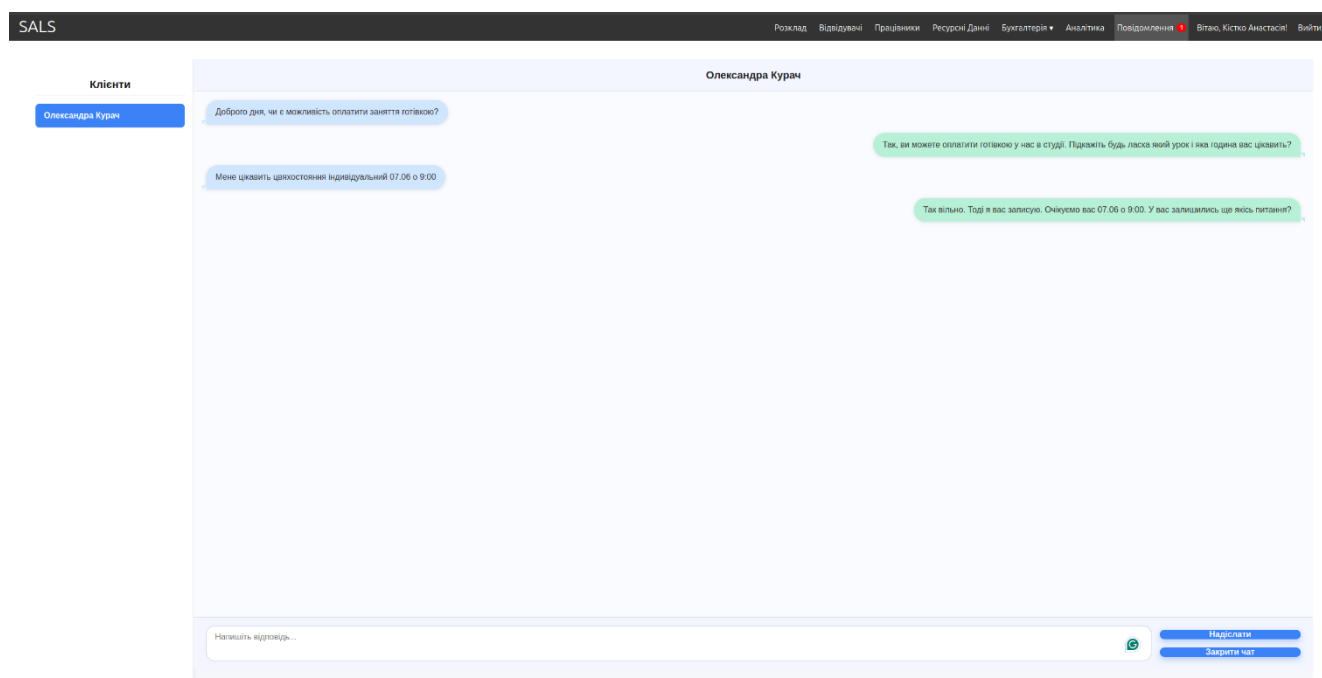


Рисунок 3.37 – Чат з відвідувачем у адміністратора

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
						67
Змн.	Арк.	№ докум.	Підпис	Дата		

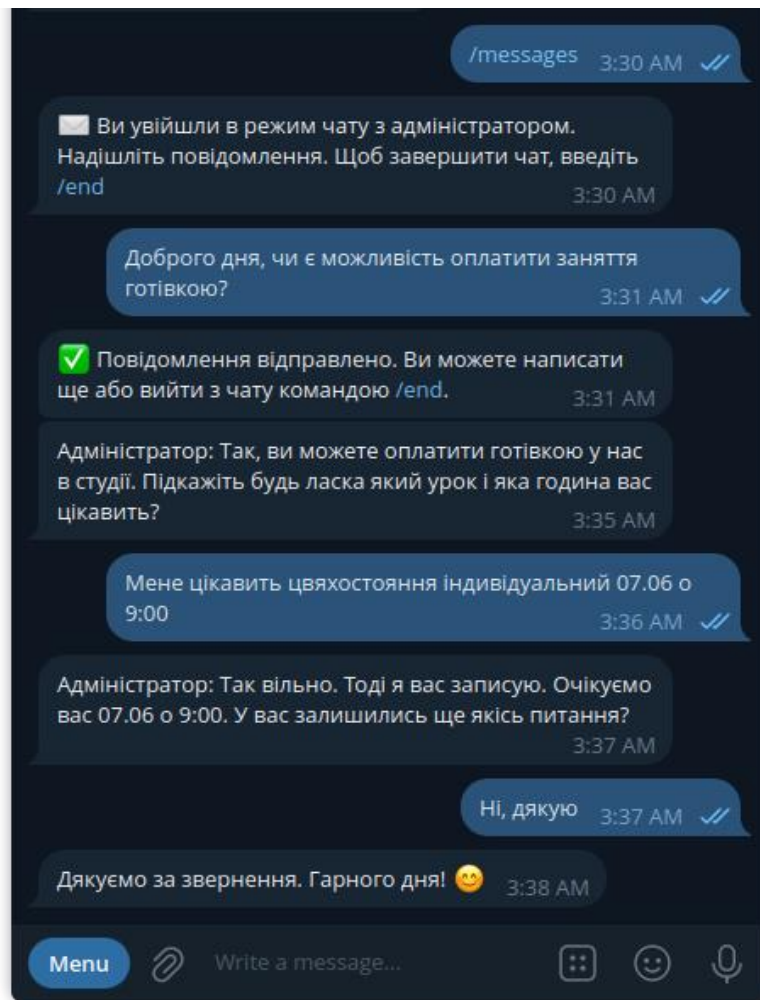


Рисунок 3.38 – Чат з адміністратором у відвідувача

3.3. Тестування роботи програмного комплексу

Через складну архітектуру проєкту та використання контейнеризації за допомогою Docker, на даному етапі розгортання повноцінного середовища на безкоштовних хостингах виявилось неможливим. Більшість безкоштовних сервісів не підтримують такий рівень складності в рамках базових тарифів, особливо коли йдеться про використання кількох сервісів (Laravel API, React, MySQL, Nginx тощо). У зв'язку з цим було обрано альтернативний підхід до перевірки ефективності вебінтерфейсу — тестування в локальному середовищі за допомогою офіційного розширення Google Lighthouse, яке інтегровано в браузер Google Chrome. Це розширення дозволяє провести аналіз якості вебсторінки без необхідності її розміщення на публічному сервері. На рис. 3.39 представлено сторінку розширення у веб-магазині chrome.

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
						68
Змн.	Арк.	№ докум.	Підпис	Дата		

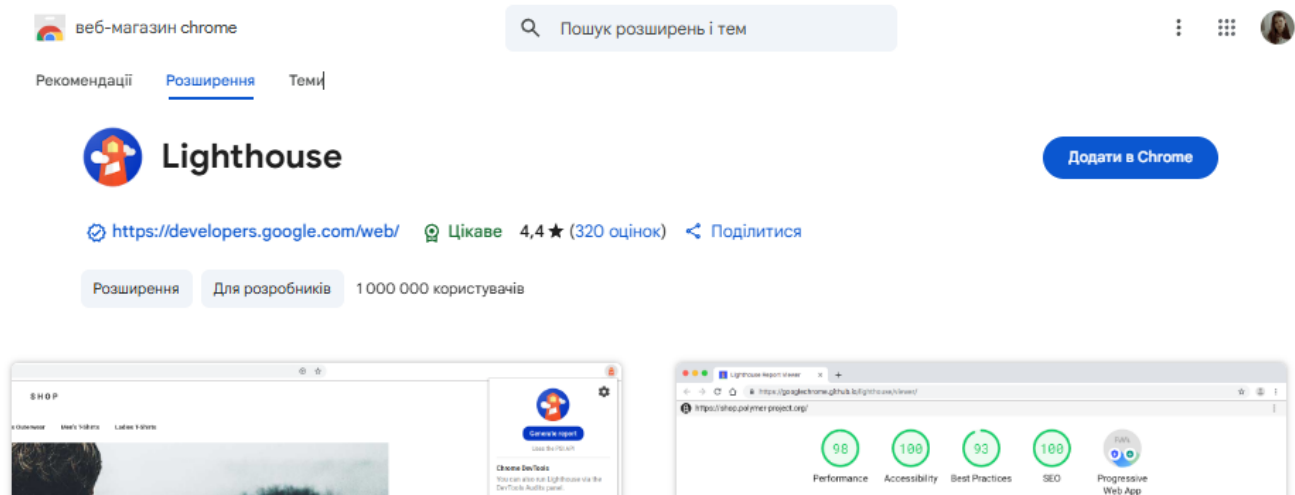


Рисунок 3.39 – Сторінка розширення у веб-магазині chrome

Для оцінки продуктивності було проаналізовано дві ключові сторінки: розклад занять (Schedule) та список клієнтів (Clients). Lighthouse надало детальні звіти щодо продуктивності, доступності, SEO та загальних рекомендацій щодо покращення, що дозволило отримати цінну інформацію для подальшої оптимізації інтерфейсу. Аналіз сторінок зображено на рис. 3.40 – 3.42.

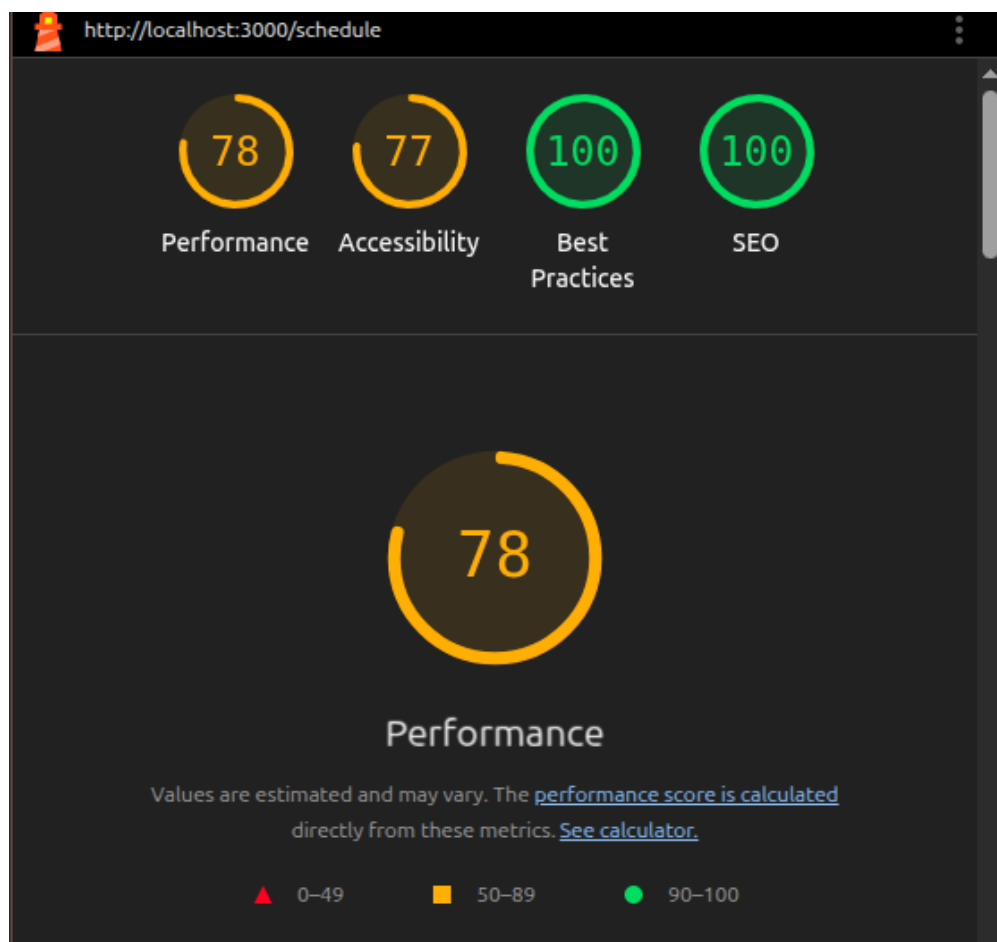


Рисунок 3.40 – Результат тестування сторінки Schedule

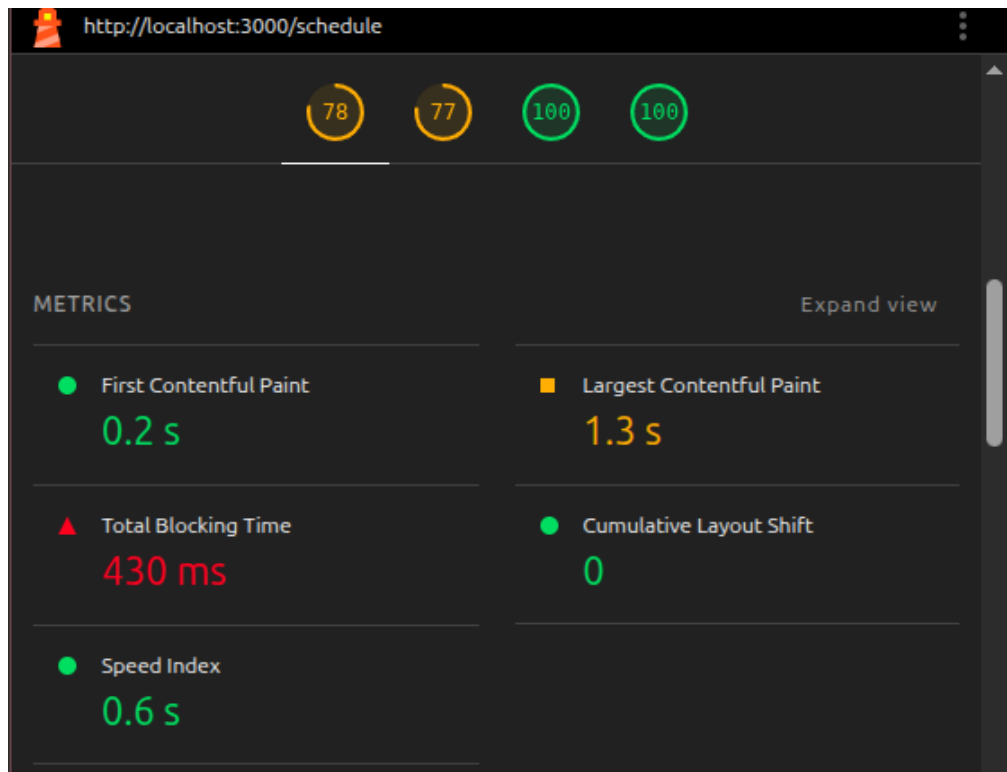


Рисунок 3.41 – Метрики тестування сторінки Schedule

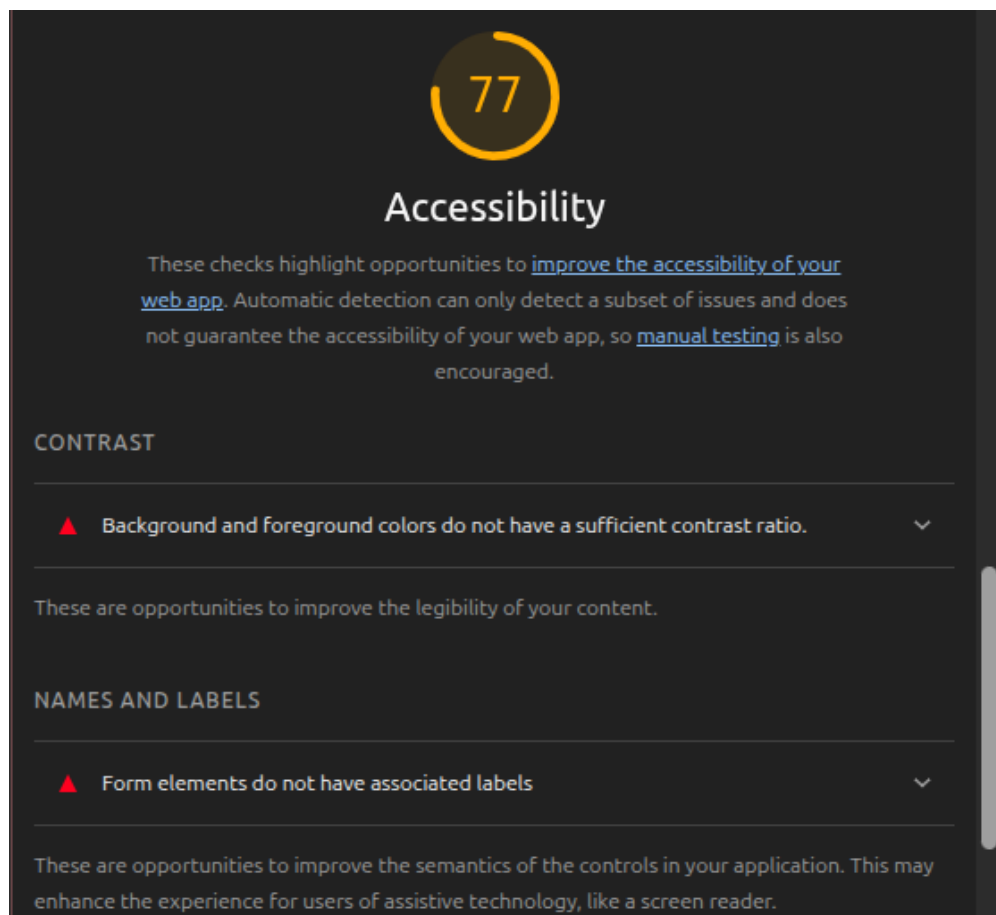


Рисунок 3.42 – Пропозиції покращення

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
						70
Змн.	Арк.	№ докум.	Підпис	Дата		

Було проаналізовано сторінку «Clients». За результатами тестування вона показала високі результати за критеріями Best Practices та SEO — по 100 балів зі 100 можливих. Це свідчить про те, що сторінка повністю відповідає сучасним вебстандартам і добре підходить для індексації в пошукових системах. Також вона не містить потенційно небезпечного коду, має коректну структуру HTML і дотримується безпечних підходів до взаємодії з браузером.

Що стосується продуктивності, сторінка отримала 81 бал. Це хороший, але не ідеальний показник. Основною причиною меншої оцінки є час блокування головного потоку — Total Blocking Time склав 350 мілісекунд. Це означає, що деякі скрипти або операції затримують завантаження і першу взаємодію з користувачем. Проте решта метрик свідчать про високу швидкість рендерингу: перший вміст з'являється вже через 0.2 секунди після завантаження, а основний вміст — через 1.3 секунди. Це дозволяє користувачу практично миттєво побачити інтерфейс після відкриття сторінки.

Іншими словами, сторінка швидко завантажується, стабільно працює, не зсуває елементи під час рендерингу (показник Layout Shift є мінімальним), але дещо потребує оптимізації JavaScript-коду для зменшення блокування.

Доступність теж отримала оцінку 81, що говорить про в цілому хорошу роботу з елементами для користувачів з особливими потребами, хоча тут ще можливі покращення. Наприклад, варто перевірити контрастність, наявність альтернативного тексту для зображень або підтримку навігації з клавіатури.

Загалом, результати тестування сторінки «Clients» свідчать про якісну реалізацію, що відповідає сучасним вебстандартам та забезпечує зручну взаємодію з користувачем. Невеликі недоліки, виявлені Lighthouse, є типовими для проєктів з великою кількістю функціональності та не становлять суттєвих проблем для реального використання. Аналіз сторінки зображено на рис. 3.43 – 3.44.

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		71

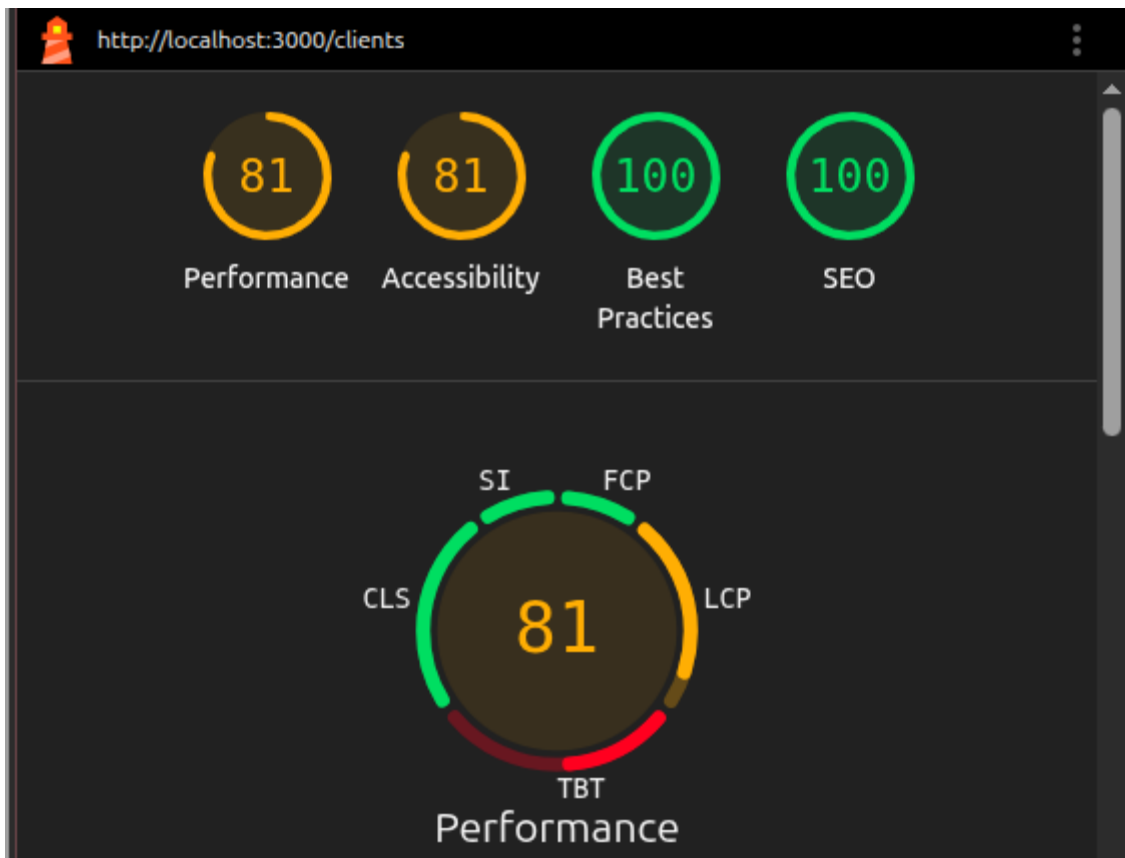


Рисунок 3.43 – Результат тестування сторінки clients

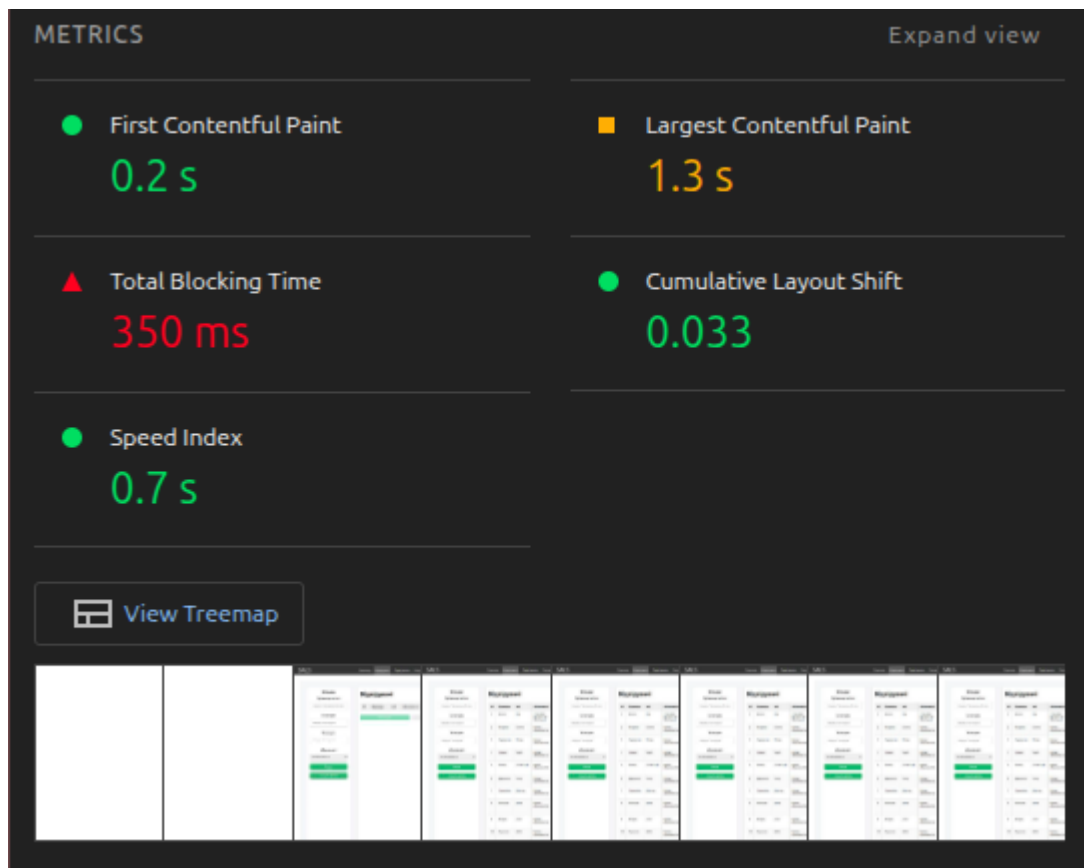


Рисунок 3.44 – Метрики тестування сторінки clients

Висновки до третього розділу

У третьому розділі було розглянуто реалізацію інтерфейсу користувача як з боку адміністратора студії, так і з боку відвідувача. Важливою особливістю проєкту є гнучкий і зручний підхід до взаємодії з системою, що було досягнуто шляхом створення повноцінного веб-інтерфейсу та Telegram-бота. Така комбінація дозволяє охопити різні сценарії використання, забезпечуючи як широкі функціональні можливості для адміністраторів, так і простоту та швидкість доступу для клієнтів.

Інтерфейс адміністратора реалізовано з використанням сучасних вебтехнологій (React, Laravel API), що дозволяє ефективно керувати студією: додавати та редагувати клієнтів, працівників, курси, проводити фінансові операції, керувати розкладом занять тощо. Усі ці можливості були протестовані, і інтерфейс продемонстрував високу стабільність і зручність у користуванні.

Інтерфейс клієнта представлено у вигляді Telegram-бота, що надає змогу переглядати розклад, записуватися на заняття, оплачувати абонементи або окремі уроки, а також переглядати історію відвідувань і платежів. Це робить користування системою максимально простим і доступним — достатньо лише мати Telegram на смартфоні.

Було також проведено тестування продуктивності системи за допомогою інструменту Lighthouse. Отримані результати свідчать про високий рівень якості розроблених сторінок: хорошу продуктивність, відповідність найкращим практикам, належну доступність і готовність до SEO-індексації. Виявлені недоліки мають незначний характер і можуть бути усунені в подальшому етапі оптимізації.

Загалом, реалізований інтерфейс відповідає усім вимогам сучасних вебзастосунків, забезпечує зручну роботу з системою для різних категорій користувачів та має потенціал для подальшого масштабування й розвитку.

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		73

ВИСНОВКИ

Під час виконання кваліфікаційної роботи бакалавра було спроектовано та реалізовано систему адміністрування навчальної студії. Проведено аналіз предметної області, досліджено основні потреби адміністративного персоналу, інструкторів та клієнтів, а також проаналізовано наявні рішення, що дозволило сформулювати перелік вимог до функціональності системи.

У процесі проектування було розроблено структуру бази даних, що охоплює облік клієнтів, працівників, курсів, розкладу занять, фінансових операцій та пов'язаних довідників. Було визначено архітектуру системи з поділом на клієнтську та серверну частини, реалізовану на основі технологій React, Laravel та Docker, що забезпечило модульність, масштабованість і зручність розгортання.

Проектування інтерфейсу користувача передбачало створення функціональних модулів для різних типів користувачів: адміністратора та клієнта. Для адміністратора було реалізовано повноцінну вебпанель для керування студією, а для клієнта — Telegram-бот, який дозволяє переглядати розклад, здійснювати записи на заняття, оплачувати послуги та переглядати історію взаємодії зі студією.

Було детально описано порядок взаємодії користувача з системою, розглянуто основні сценарії використання, а також забезпечено логічну послідовність виконання дій. Особливу увагу приділено реалізації розкладу занять із можливістю drag-and-drop, створенням і редагуванням уроків, що забезпечує гнучке планування навчального процесу.

Таким чином, у результаті виконання кваліфікаційної роботи було створено повноцінну систему для адміністрування навчальної студії, яка охоплює всі ключові процеси її функціонування. Реалізоване рішення є ефективним, зручним у використанні та готовим до впровадження в реальних умовах.

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		74

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. A Practical Introduction to Laravel Eloquent ORM. DigitalOcean. URL: <https://www.digitalocean.com/community/tutorial-series/a-practical-introduction-to-laravel-eloquent-orm>.
2. Axios - Promise based HTTP client for the browser and node.js. axios - npm. URL: <https://www.npmjs.com/package/axios#features>.
3. Controllers. Laravel. URL: <https://laravel.com/docs/11.x/controllers>.
4. Cooper A. About Face: The Essentials of User Interface Design. Wiley, 1995.
5. CRM Dance. URL: <https://crm.dance>.
6. Docker Documentation. URL: <https://docs.docker.com/>.
7. EasyBusy. URL: <https://easy-busy.cloud/>.
8. Introduction to Lighthouse. Chrome for developers. URL: <https://developer.chrome.com/docs/lighthouse/overview>.
9. LiqPay - APIs. URL: <https://www.liqpay.ua/doc/api>.
10. Model Classes. Laravel. URL: <https://laravel.com/docs/12.x/eloquent>.
11. NeoDove. URL: <https://neodove.com>.
12. Nginx Documentation. URL: <https://nginx.org/en/docs/>.
13. OneBox OS. URL: <https://1b.app/ua/crm-for/yoga-studio/>.
14. OtmechalkaCRM. URL: <https://otmechalka.com>.
15. React - The library for web and native user interfaces. URL: <https://reactjs.org>.
16. Telegram Bot API. URL: <https://core.telegram.org/bots/api>.
17. The PHP Framework for Web Artisans. Laravel. URL: <https://laravel.com/>.
18. Архітектура вебзастосунків. R_D media. URL: <https://robotdreams.cc/uk/blog/567-arhitektura-vebzastosunkiv>.
19. АРХІТЕКТУРНІ ПІДХОДИ ДО РОЗРОБКИ МАСШТАБОВАНИХ ВЕБ-ЗАСТОСУНКІВ / О. Складенко та ін. Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка», 2024. URL: <https://doi.org/10.28925/2663-4023.2024.24.341350>.
20. Вступ до REST API — RESTful вебсервіси. R_D media. URL: <https://robotdreams.cc/uk/blog/466-vstup-do-rest-api-restful-vebservisi>.

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
						75
Змн.	Арк.	№ докум.	Підпис	Дата		

21. Гуцуляк О. Порівнюємо React, Angular і Vue — найпопулярніші бібліотеки й фреймворки. DOU. URL: <https://dou.ua/forums/topic/39933/>.
22. Каграманова Ю. Як будувати UML-діаграми.. DOU. URL: <https://dou.ua/forums/topic/40575/>.
23. Кравчук О. А. Огляд та аналіз архітектури прогресивних веб-додатків. 2017. 132–136 с.
24. Ознайомлення з патерном MVC (Model-View-Controller). JavaRush. URL: <https://javarush.com/ua/groups/posts/uk.2536.chastina-7-oznayomlennja-z-paternom-mvc-model-view-controller>.
25. Переваги фреймворку Laravel для веб-розробки. Wezom. URL: <https://wezom.com.ua/ua/blog/17-preimuschestv-ispolzovanija-laravel-v-it-industrii>.
26. Проектування баз даних. URL: https://dSPACE.wunu.edu.ua/bitstream/316497/41356/1/FCIT_kKN_bIPZ_dPBD_LEC.pdf.
27. Редакція Академії ITSTEP. Введення в UI дизайн: створення ефективних та привабливих інтерфейсів. ITSTEP. URL: <https://cloud.itstep.org/blog/introduction-to-ui-design-creating-effective-and-attractive-interfaces>.
28. Ситник Н. В. Проектування баз і сховищ даних: Навчальний посібник. К. : КНЕУ, 2019.
29. Що таке MVP і як створити MVP продукт. Wezom. URL: <https://wezom.com.ua/ua/blog/chto-takoe-mvp>.
30. Як інтегрувати платіжні системи для підвищення фінансової ефективності. PaySaxas. URL: <https://paysaxas.com/uk/yak-integruvaty-platizhni-systemy-dlya-p/>.

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		76

ДОДАТКИ

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
						77
Змн.	Арк.	№ докум.	Підпис	Дата		

```

<?php

namespace App\Http\Controllers;

use App\Models\Lesson;
use Illuminate\Http\Request;

class LessonController extends Controller
{
    // Отримати всі уроки
    public function index(Request $request)
    {
        $query = Lesson::with('lessonType', 'course', 'instructor.employee',
        'studio');

        // Якщо передано start_date i end_date - фільтруємо
        if ($request->has('start_date') && $request->has('end_date')) {
            $startDate = $request->input('start_date');
            $endDate = $request->input('end_date');

            // Перевірка формату дати не є обов'язковою, але бажана для надійності
            $query->whereBetween('start_time', [$startDate . ' 00:00:00', $endDate
            . ' 23:59:59']);
        }

        $lessons = $query->get();

        return response()->json([
            'lessons' => $lessons
        ]);
    }

    public function show($id)
    {
        $lesson = Lesson::with('lessonType', 'course', 'instructor.employee',
        'studio')->find($id);

        if (!$lesson) {
            return response()->json(['message' => 'Працівника не знайдено'], 404);
        }

        return response()->json($lesson);
    }

    public function store(Request $request)
    {
        $validated = $request->validate([
            'name' => 'required|string',
            'lesson_type_id' => 'required|exists:lesson_types,id',
            'course_id' => 'required|exists:courses,id',
            'instructor_id' => 'required|exists:instructors,id',
            'clients' => 'required|string',
            'start_time' => 'required|date',
            'end_time' => 'required|date|after:start_time',
            'price' => 'required|numeric',
            'studio_id' => 'required|exists:studios,id',
            'status' => 'required|string',
        ]);
    }
}

```

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		78

```

    $lesson = Lesson::create($validated);

    $lesson->load('lessonType', 'course', 'instructor', 'studio');

    return response()->json($lesson);
}

public function update(Request $request, $id)
{
    $lesson = Lesson::findOrFail($id);

    $validated = $request->validate([
        'name' => 'required|string',
        'lesson_type_id' => 'required|exists:lesson_types,id',
        'course_id' => 'required|exists:courses,id',
        'instructor_id' => 'required|exists:instructors,id',
        'clients' => 'required|string',
        'start_time' => 'required|date',
        'end_time' => 'required|date|after:start_time',
        'price' => 'required|numeric',
        'studio_id' => 'required|exists:studios,id',
        'status' => 'required|string',
    ]);

    $lesson->update($validated);

    $lesson->load('lessonType', 'course', 'instructor', 'studio');

    return response()->json($lesson);
}

// Видалити урок
public function destroy(Lesson $lesson)
{
    $lesson->delete();
    return response()->json(['message' => 'Lesson deleted']);
}
}

```

```

<?php

namespace App\Http\Controllers;

use App\Models\Employee;
use App\Models\Instructor;
use Illuminate\Http\Request;
use App\Models\CourseInstructor;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Facades\Log;

class EmployeeController extends Controller
{
    // Повертає список працівників з пагінацією і фільтрами
    public function index(Request $request)
    {
        $query = Employee::with([
            'instructor.courses',
            'instructor.rates.lessonType',
            'instructor.rates.course',
            'studioAdmin'
        ]);

        // Фільтри (за позицією, курсом, телеграмом, ім'ям)
        if ($request->filled('position')) {
            $query->where('position', $request->input('position'));
        }

        if ($request->filled('telegram')) {
            $query->whereHas('instructor', function ($q) use ($request) {
                $q->where('telegram', 'like', '%' . $request->input('telegram') .
'');
            });
        }

        if ($request->filled('name')) {
            $name = $request->input('name');
            $query->where(function($q) use ($name) {
                $q->where('first_name', 'like', "%$name%")
                    ->orWhere('last_name', 'like', "%$name%");
            });
        }

        if ($request->filled('course')) {
            $courseId = (int) $request->input('course');
            $query->whereHas('instructor.courses', function ($q) use ($courseId) {
                $q->where('courses.id', $courseId);
            });
        }

        if ($request->filled('status')) {
            $status = $request->input('status');
            $query->where('status', $status);
        }

        $employees = $query->paginate(10);
    }
}

```

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
						80
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        return response()->json($employees);
    }

    // Показ одного працівника
    public function show($id)
    {
        $employee = Employee::with([
            'instructor.courses',
            'instructor.rates.course',
            'instructor.rates.lessonType',
            'studioAdmin'
        ])->find($id);

        if (!$employee) {
            return response()->json(['message' => 'Працівника не знайдено'], 404);
        }

        return response()->json($employee);
    }

    // Створення працівника
    public function store(Request $request)
    {
        $validated = $request->validate([
            'first_name' => 'required|string|max:255',
            'last_name' => 'required|string|max:255',
            'phone_number' => 'required|string|max:255',
            'position' =>
                'required|string|in:адміністратор,інструктор,прибиральник,бухгалтер,охоронець',
            'salary' => 'required|numeric',
            'status' => 'required|string|in:працює,звільнений',
            'telegram' => 'nullable|string|max:255',
            'courses' => 'nullable|string', // рядок з ID курсів "1,2,3"
            'password' => 'nullable|string|min:6',
        ]);

        DB::beginTransaction();

        try {
            $employee = Employee::create($validated);

            if ($employee->position === 'адміністратор') {
                $password = $request->input('password', 'default_password');
                $employee->studioAdmin()->updateOrCreate([], [
                    'username' => $request->input('username', 'default'),
                    'password' => Hash::make($password),
                ]);
            }

            if ($employee->position === 'інструктор') {
                $instructor = Instructor::create([
                    'telegram' => $request->input('telegram', ''),
                    'employee_id' => $employee->id,
                ]);

                if ($request->filled('courses')) {
                    $courseIds = array_filter(array_map('intval', explode(',', $request->input('courses'))));
                    $instructor->courses()->sync($courseIds);
                }
            }
        }
    }

```

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
						81
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        DB::commit();

        $employee->load('instructor.courses', 'studioAdmin');

        return response()->json($employee, 201);
    } catch (\Exception $e) {
        DB::rollBack();
        return response()->json(['message' => 'Помилка при створенні
працівника', 'error' => $e->getMessage()], 500);
    }
}

// Оновлення працівника
public function update(Request $request, $id)
{
    $employee = Employee::find($id);

    if (!$employee) {
        return response()->json(['message' => 'Працівника не знайдено'], 404);
    }

    $validated = $request->validate([
        'first_name' => 'sometimes|required|string|max:255',
        'last_name' => 'sometimes|required|string|max:255',
        'phone_number' => 'sometimes|required|string|max:255',
        'position' =>
'required|string|in:адміністратор,інструктор,прибиральник,бухгалтер,охоронець',
        'salary' => 'sometimes|required|numeric',
        'status' => 'required|string|in:працює,звільнений',
        'telegram' => 'nullable|string|max:255',
        'username' => 'nullable|string|max:255',
        'password' => 'nullable|string|min:6',
        'courses' => 'nullable|array',
        'courses.*' => 'integer|exists:courses,id',
        'instructor' => 'nullable|array',
        'instructor.courses' => 'nullable|array',
        'instructor.courses.*' => 'integer|exists:courses,id',
        'instructor.rates' => 'nullable|array',
        'instructor.rates.*.lesson_type_id' =>
'required|integer|exists:lesson_types,id',
        'instructor.rates.*.course_id' => 'required|integer|exists:courses,id',
        'instructor.rates.*.rate' => 'required|numeric',
    ]);
    Log::info('Отриманий запит на оновлення працівника', $request->all());
    DB::beginTransaction();

    try {
        // Оновлюємо основні дані працівника
        $employee->update($validated);

        if ($employee->position === 'адміністратор') {
            $adminData = ['username' => $validated['username'] ?? 'default'];

            // Якщо пароль передано, то оновлюємо його з хешуванням
            if (!empty($validated['password'])) {
                $adminData['password'] = Hash::make($validated['password']);
            }

            $employee->studioAdmin()->updateOrCreate([], $adminData);
        }
    }
}

```

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
						82
Змн.	Арк.	№ докум.	Підпис	Дата		

```

// Видаляємо інструктора, якщо є
$instructor = Instructor::where('employee_id', $employee->id)-
>first();

if ($instructor) {
    $instructor->courses()->detach();
    $instructor->delete();
}
} else if ($employee->position === 'інструктор') {
    Log::info('Початок оновлення інструктора', [
        'employee_id' => $employee->id,
        'validated' => $validated,
    ]);

// Оновлюємо або створюємо інструктора
$instructor = Instructor::firstOrCreate(['employee_id' => $employee-
>id]);

$instructor->telegram = $validated['telegram'] ?? '';
$instructor->employee_id = $employee->id;
$instructor->save();

Log::info('Інструктор після збереження', [
    'instructor_id' => $instructor->id,
    'telegram' => $instructor->telegram,
]);

if (isset($validated['instructor']['rates'])) {
    $rates = $validated['instructor']['rates'];
    Log::info('Отримані ставки', ['rates' => $rates]);

    // Витягаємо унікальні course_id з ставок
    $courseIds = collect($rates)->pluck('course_id')->unique()-
>toArray();

    Log::info('Унікальні course_id для синхронізації',
['course_ids' => $courseIds]);

    $instructor->courses()->sync($courseIds);

    // Видаляємо старі ставки
    Log::info('Видалення старих ставок інструктора');
    $instructor->rates()->delete();

    // Додаємо нові ставки
    Log::info('Додавання нових ставок', ['rates' => $rates]);
    foreach ($rates as $rateData) {
        $instructor->rates()->create($rateData);
    }
}

if ($employee->studioAdmin) {
    Log::info('Видалення студійного адміністратора', [
        'studio_admin_id' => $employee->studioAdmin->id,
    ]);
    $employee->studioAdmin()->delete();
}

Log::info('Оновлення інструктора завершено', ['instructor_id' =>
$instructor->id]);
} else {
    // Для інших позицій видаляємо інструктора і курси, якщо існують
    $instructor = Instructor::where('employee_id', $employee->id)-
>first();

```

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
						83
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        if ($instructor) {
            $instructor->courses()->detach();
            $instructor->delete();
        }
        // Також видаляємо адміністратора, якщо є
        if ($employee->studioAdmin) {
            $employee->studioAdmin()->delete();
        }
    }

    DB::commit();

    $employee->load('instructor.courses', 'studioAdmin');

    return response()->json($employee);

} catch (\Exception $e) {
    DB::rollBack();
    return response()->json([
        'message' => 'Помилка при оновленні працівника',
        'error' => $e->getMessage()
    ], 500);
}

}

// Видалення працівника
public function destroy($id)
{
    $employee = Employee::find($id);

    if (!$employee) {
        return response()->json(['message' => 'Працівника не знайдено'], 404);
    }

    $employee->delete();

    return response()->json(['message' => 'Працівника видалено']);
}
}

```

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		84

```

return (
  <div
    className="scheduler"
    onMouseUp={handleMouseUp} // відловлюємо mouseup на батьківському
    контейнері
  >
    <div style={{ marginBottom: '10px' }}>
      <strong>Вибраний тиждень: </strong>{weekRange}
    </div>
    <div className={'scheduler-pagination'} style={{ marginBottom: '10px',
display: 'flex', alignItems: 'center', gap: '10px' }}>
      <button onClick={goToPreviousWeek}>← Попередній тиждень</button>
      <input
        type="date"
        value={currentMonday.toISOString().slice(0,10)}
        onChange={handleDateChange}
      />
      <button onClick={goToNextWeek}>Наступний тиждень →</button>
    </div>
    <div className="header-row">
      <div className="time-cell"></div>
      {days.map(day => <div className="day-header" key={day}>{day}</div>)}
    </div>
    <div className="grid" style={{ position: "relative", height:
`${hours.length * 60}px` }}>
      {hours.map(hour => (
        <div className="row" key={hour}>
          <div className="time-cell">{hour}:00</div>
          {days.map((_, dayIndex) => (
            <div
              key={dayIndex + "-" + hour}
              className={`cell ${isCellSelected(dayIndex, hour) ?
'selected' : ''}`}
              onMouseDown={() => handleMouseDown(dayIndex, hour)}
              onMouseEnter={() => handleMouseEnter(dayIndex, hour)}
              // прибираємо onClick, щоб не відкривалось окреме вікно
              при кліку
            ></div>
          ))}
        </div>
      ))}
    </div>
  ))}

  {Object.entries(lessonsByDay).map(([dayStr, dayLessons]) => {
    const day = Number(dayStr);
    const groups = groupOverlappingLessons(dayLessons);

    return groups.map(group => {
      return group.map((lesson, index) => {
        const start = new Date(lesson.start_time);
        const end = new Date(lesson.end_time);
        const top = (start.getHours() - 8) * 60 +
start.getMinutes();

        const height = (end - start) / 60000;
        const dayLeftPercent = (day * 100) / 7;

        const widthPercent = 100 / 7 / group.length;
        const leftPercent = dayLeftPercent + index * widthPercent;

        return (
          <div
            key={lesson.id}

```

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
						85
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        className="lesson-block"
        style={{
            top: `${top}px`,
            height: `${height}px`,
            width: `calc(${widthPercent}% - 4px)`,
            left: `calc(${leftPercent}% + 60px)`,
            backgroundColor: lesson.status === "cancelled"
                ? '#981b2e'
                : lesson.status === "scheduled"
                    ? '#d6ab1b' // СВІТЛО-ЖОВТИЙ ФОН
                    : '',
            position: "absolute",
            cursor: "pointer",
        }}
        onClick={() => handleLessonClick(lesson)}
        title={`Урок:
        ${lesson.name}\n${start.toLocaleTimeString([], { hour: '2-digit', minute: '2-
        digit' })}-${end.toLocaleTimeString([], { hour: '2-digit', minute: '2-digit' })}`}
        >
        <strong>{lesson.name}</strong><br />
        {start.toLocaleTimeString([], { hour: '2-digit',
        minute: '2-digit' })}-${end.toLocaleTimeString([], { hour: '2-digit', minute: '2-
        digit' })}
        </div>
    );
    });
    });
</div>

{selectedLesson && (
    <div className="modal-overlay" onClick={() => setSelectedLesson(null)}>
        <div className="modal-container" onClick={e =>
e.stopPropagation()}>
            <button className="close-button" onClick={() =>
setSelectedLesson(null)}>&times;</button>
            <h3>Деталі уроку</h3>

            <p><strong>Назва:</strong> {selectedLesson.name}</p>
            <p><strong>Початок:</strong> {new
Date(selectedLesson.start_time).toLocaleString()}</p>
            <p><strong>Кінець:</strong> {new
Date(selectedLesson.end_time).toLocaleString()}</p>
            <p><strong>Ціна:</strong> {selectedLesson.price}</p>
            <p><strong>Тип уроку:</strong>
{selectedLesson.lesson_type?.type || 'Немає'}</p>
            <p><strong>Курс:</strong> {selectedLesson.course?.name ||
'Немає'}</p>
            <p><strong>Інструктор:</strong>
{selectedLesson.instructor.employee?.first_name}
{selectedLesson.instructor.employee?.last_name}</p>

            <h4>Клієнти</h4>
            {selectedLesson.clients &&
Array.isArray(selectedLesson.clients) && selectedLesson.clients.length > 0 ? (
                <table className="clients-table">
                    <thead>
                        <tr>
                            <th>Ім'я</th>
                            <th>Прізвище</th>
                            <th>Телефон</th>
                            <th>Телеграм</th>
                        </tr>

```

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
						86
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        </thead>
        <tbody>
        {selectedLesson.clients.map(clientId => {
            const client = allClients.find(c => c.id ===
clientId);

            if (!client) return null;
            return (
                <tr key={client.id}>
                    <td>{client.first_name}</td>
                    <td>{client.last_name}</td>
                    <td>{client.phone}</td>
                    <td>{client.telegram}</td>
                </tr>
            );
        })}
        </tbody>
    </table>
    ) : (
        <p>Немає клієнтів</p>
    )}

    <p><strong>Студія:</strong> {selectedLesson.studio?.name}</p>
    <p><strong>Статус:</strong> {
        LESSON_STATUSES.find(s => s.value ===
selectedLesson?.status)?.label || '—'
    }</p>

    <div className="form-buttons">
        <button onClick={() => {
            setEditLesson(selectedLesson);
            setSelectedLesson(null);
            setShowForm(true);
            setFormData({
                ...selectedLesson,
                clients: selectedLesson.clients || []
            });
        }}>Редагувати</button>

        <button className={'duplicate-button'} onClick={() => {
            setShowDuplicatePopup(true);
            setDuplicateSettings({
                lesson: selectedLesson,
                repeatCount: 1,
                selectedDays: [],
                time: selectedLesson.start_time.slice(11, 16),
            });
        }}>Дублювати</button>

        <button className={'cancel-button'} onClick={() =>
setSelectedLesson(null)}>Закрити</button>
    </div>
</div>
</div>

    )}

    {showDuplicatePopup && (
        <div className="modal-overlay" onClick={() =>
setShowDuplicatePopup(false)}>
            <div className="modal" onClick={e => e.stopPropagation()}>
                <h3>Дублювання уроку: {duplicateSettings.lesson.name}</h3>

```

```

<label>Кількість повторень</label>
<input
  type="number"
  min={1}
  value={duplicateSettings.repeatCount}
  onChange={e => setDuplicateSettings({ ...duplicateSettings,
repeatCount: parseInt(e.target.value) })}
/>

<label>Оберіть дні тижня</label>
<div className="days-checkboxes">
  {["Пн", "Вт", "Ср", "Чт", "Пт", "Сб", "Нд"].map((dayName,
idx) => (
    <label key={idx}>
      <input
        type="checkbox"
        checked={duplicateSettings.selectedDays.include
s(idx)}
        onChange={() => {
          const isSelected =
duplicateSettings.selectedDays.includes(idx);
          setDuplicateSettings({
            ...duplicateSettings,
            selectedDays: isSelected
?
duplicateSettings.selectedDays.filter(d => d !== idx)
:
[...duplicateSettings.selectedDays, idx],
          });
        }}
      />
      {dayName}
    </label>
  ))}
</div>

<label>Час початку</label>
<input
  type="time"
  value={duplicateSettings.time}
  onChange={e => setDuplicateSettings({ ...duplicateSettings,
time: e.target.value })}
/>

<button onClick={handleConfirmDuplicate}>Підтверджую
дублювання</button>
<button onClick={() =>
setShowDuplicatePopup(false)}>Скасувати</button>
</div>
</div>
)}

{showForm && (
  <div className="modal-overlay" onClick={() => setShowForm(false)}>
    <div className="modal-container" onClick={e =>
e.stopPropagation()}>
      <button className="close-button" onClick={() =>
setShowForm(false)}>
        &times;
      </button>
      <h3 className="modal-title">
        {formData.id ? "Редагувати урок" : "Новий урок"}
      </h3>
    </div>
  </div>
)}

```

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
						88
Змн.	Арк.	№ докум.	Підпис	Дата		

```

</h3>

<form onSubmit={handleSubmit} className="modal-form">
  <div className="form-group">
    <label>Назва:</label>
    <input
      type="text"
      value={formData.name}
      onChange={ (e) => setFormData({ ...formData, name:
e.target.value })}
      required
    />
  </div>

  <div className="form-row">
    <div className="form-group">
      <label>Початок:</label>
      <input
        type="datetime-local"
        value={formatDateForInput(formData.start_time)}
        onChange={ (e) => setFormData({ ...formData,
start_time: e.target.value })}
        required
      />
    </div>

    <div className="form-group">
      <label>Кінець:</label>
      <input
        type="datetime-local"
        value={formatDateForInput(formData.end_time)}
        onChange={ (e) => setFormData({ ...formData,
end_time: e.target.value })}
        required
      />
    </div>
  </div>

  <div className="form-group">
    <label>Ціна:</label>
    <input
      type="number"
      value={formData.price}
      onChange={ (e) => setFormData({ ...formData, price:
e.target.value })}
    />
  </div>

  <div className="form-row">
    <div className="form-group">
      <label>Тип уроку:</label>
      <select
        value={formData.lesson_type_id}
        onChange={ (e) => setFormData({ ...formData,
lesson_type_id: e.target.value })}
        required
      >
        <option value="">Виберіть тип</option>
        {allLessonTypes.map((l) => (
          <option key={l.id} value={l.id}>
            {l.type}
          </option>
        ))}
      </select>
    </div>
  </div>
</form>

```

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
						89
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    ))}
  </select>
</div>

<div className="form-group">
  <label>Курс:</label>
  <select
    value={formData.course_id}
    onChange={ (e) => setFormData({ ...formData,
course_id: e.target.value })}
    required
  >
    <option value="">Виберіть курс</option>
    {allCourses.map((c) => (
      <option key={c.id} value={c.id}>
        {c.name}
      </option>
    ))}
  </select>
</div>

<div className="form-group">
  <label>Інструктор:</label>
  <select
    value={formData.instructor_id}
    onChange={ (e) => setFormData({ ...formData,
instructor_id: e.target.value })}
    required
  >
    <option value="">Виберіть інструктора</option>
    {allInstructors.map((instr) => (
      <option key={instr.id} value={instr.id}>
        {instr.employee.first_name}
        {instr.employee.last_name}
      </option>
    ))}
  </select>
</div>

<div className="form-group">
  <label>Клієнти:</label>
  <Select
    options={allClients.map((client) => ({
      value: client.id,
      label: `${client.first_name}
${client.last_name} (${client.phone || client.telegram || ""})`,
    }))}
    isMulti
    value={allClients
      .filter((client) =>
formData.clients.includes(client.id))
      .map((client) => ({
        value: client.id,
        label: `${client.first_name}
${client.last_name} (${client.phone || client.telegram || ""})`,
      }))}
    onChange={(selectedOptions) => {
      if (selectedOptions.length <= 20) {
        setFormData({
          ...formData,
          clients: selectedOptions.map((opt) =>
opt.value),

```

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
						90
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        });
        } else {
            alert("Можна вибрати не більше 20
клієнтів");
        }
    }}
    placeholder="Виберіть клієнтів"
    closeMenuOnSelect={false}
    classNamePrefix="react-select"
  />
</div>

<div className="form-row">
  <div className="form-group">
    <label>Студія:</label>
    <select
      value={formData.studio_id}
      onChange={ (e) => setFormData({ ...formData,
studio_id: e.target.value })}
      required
    >
      <option value="">Виберіть студію</option>
      {allStudios.map((s) => (
        <option key={s.id} value={s.id}>
          {s.name}
        </option>
      ))}
    </select>
  </div>

  <div className="form-group">
    <label>Статус:</label>
    <select
      value={formData.status}
      onChange={ (e) => setFormData({ ...formData,
status: e.target.value })}
    >
      {LESSON_STATUSES.map((status) => (
        <option key={status.value}
value={status.value}>
          {status.label}
        </option>
      ))}
    </select>
  </div>
</div>

<div className="form-buttons">
  <button type="submit" className="submit-button">
    Зберегти
  </button>
  <button type="button" className="cancel-button"
onClick={ () => setShowForm(false) }>
    Закрити
  </button>
</div>
</form>
</div>
</div>
  )}
</div>
);

```

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
						91
Змн.	Арк.	№ докум.	Підпис	Дата		

					ІПЗ.КР.Б-121-25-ПЗ	Арк.
						92
Змн.	Арк.	№ докум.	Підпис	Дата		