МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інженерії програмного забезпечення

_____ Євген ДАВИДЕНКО

«19» червня 2026 р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

CRM СИСТЕМА ДЛЯ СТВОРЕННЯ ВІЗИТІВ

Спеціальність 121 Інженерія програмного забезпечення

Освітня програма «Інженерія програмного забезпечення»

Здобувач _____ **Олександр ГОРДІЄНКО**

«19» червня 2026 р.

Керівник роботи

PhD, доцентка (6. в. з.) _____ **Катерина АНТІПОВА**

«19» червня 2026 р.

Миколаїв – 2026

Завдання на виконання кваліфікаційної роботи

Чорноморський національний університет імені Петра Могили

Факультет	Комп'ютерних наук
Кафедра	Інженерії програмного забезпечення
Рівень вищої освіти	Перший (бакалаврський)
Освітній ступінь	Бакалавр
Спеціальність	121 Інженерія програмного забезпечення
Освітня програма	Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри інженерії
програмного забезпечення
_____ Євген ДАВИДЕНКО
«28» грудня 2025 р.

ЗАВДАННЯ
на кваліфікаційну бакалаврську роботу здобувача
Гордієнко Олександр

1. Тема кваліфікаційної роботи CRM система для створення візитів затверджена наказом ректора ЧНУ ім. Петра Могили №349 від «26» грудня 2026р.

2. Строк представлення кваліфікаційної роботи «22» червня 2026р.

3. Результатом роботи є розроблена CRM-система для створення візитів. Початковими даними є вимоги до функціональності системи, структура предметної області, а також моделі взаємодії користувачів (клієнтів, майстрів, адміністраторів).

4. Перелік питань, що підлягають розробц

- Аналіз предметної області;
- Визначення вимог до системи;
- Проєктування структури бази даних;
- Розробка логічної та фізичної моделі БД;
- Реалізація CRM-системи;
- Тестування та перевірка працездатності системи;
- Розробка рекомендацій щодо використання системи.

5. Перелік графічних матеріалів:

- Схема бази даних (ER-діаграма);
- UML-діаграми (use case, class diagram);
- Структура таблиць бази даних;
- Схеми взаємодії користувачів із системою;
- Презентація
- Скріншоти роботи програмного забезпечення.

6. Консультанти:

Консультант	**Кафедра (організація)**	**Частина роботи**

Дата видачі завдання «28» грудня 2025р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема: **CRM система для стоврення візитів**

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КБР	26.12.2025	28.12.2025	Виконано
2.	Огляд літератури за темою роботи	03.02.2026	12.02.2026	Виконано
3.	Складання календарного плану КБР	23.03.2026	23.03.2026	Виконано
4.	Аналіз предметної області	24.03.2026	25.03.2026	Виконано
5.	Розробка проєктних рішень	26.02.2026	28.03.2026	Виконано
6.	Моделювання та конструювання ПЗ	30.03.2026	04.04.2026	Виконано
7.	Кодування, тестування та апробація розробленого ПЗ, аналіз результатів тестування, розробка керівництва користувача	10.04.2026	03.05.2026	Виконано
8.	Відгук керівника КБР	09.05.2026	09.05.2026	Виконано
9.	Оформлення КБР та презентації	13.05.2026	26.05.2026	Виконано
10.	Попередній захист	27.05.2026	27.05.2026	Виконано
11.	Рецензування	28.05.2026	28.05.2026	Виконано
12.	Завершення оформлення КБР та презентації	01.06.2026	19.06.2026	Виконано
13.	Захист кваліфікаційної роботи	23.06.2026	23.06.2026	Виконано

Здобувач _____ **Олександр ГОРДІЄНКО**

«23» березня 2026 р.

Керівник роботи

PhD, (б. в. з.)

доцентка _____ **Катерина АНТІПОВА**

«23» березня 2026р.

АНОТАЦІЯ

до кваліфікаційної бакалаврської роботи

CRM система для створення візитів

Здобувач 408 гр.: Гордієнко Олександр

Керівник: PhD, (б. в. з.) доцентка

Антіпова Катерина Олександрівна

Актуальність теми обумовлена необхідністю автоматизації процесів управління клієнтами та організації роботи салонів краси, що дозволяє підвищити ефективність обслуговування та оптимізувати використання ресурсів.

Об’єктом роботи є процеси планування та організації роботи CRM системи створення візитів.

Предметом роботи є інструменти розробки CRM-системи.

Метою роботи є розробка системи для забезпечення ефективної взаємодії між клієнтами, адміністраторами та працівниками.

Кваліфікаційна робота складається із вступу, 4 розділів, висновків та переліку джерел посилання.

У вступі обґрунтовано актуальність теми, визначено мету, завдання, об’єкт та предмет дослідження, а також описано практичне значення отриманих результатів.

У першому розділі проведено аналіз предметної області, розглянуто особливості функціонування системи, досліджено існуючі CRM-системи, їх переваги та недоліки. Також визначено основні вимоги до майбутньої системи та сформовано технічне завдання.

Другий розділ присвячено проєктуванню CRM-системи. Описано архітектуру застосунку, обґрунтовано вибір технологій, розроблено структуру бази даних, діаграми класів та взаємодії. Визначено основні модулі системи та їх функціональність.

У третьому розділі реалізовано програмну частину системи. Описано процес розробки, інтерфейс користувача, функціональні можливості (керування

клієнтами, запис на послуги, облік працівників, аналітика). Проведено тестування системи та оцінку її ефективності.

У четвертому розділі було виконано розробку та програмну реалізацію CRM-системи управління візитами для підприємств сфери послуг. Під час виконання роботи спроєктовано реляційну базу даних у середовищі Microsoft SQL Server, що забезпечує зберігання та обробку інформації про користувачів, філії, майстрів, клієнтів, послуги, візити та фінансові операції. На основі мови програмування PHP реалізовано серверну частину системи, включаючи модулі реєстрації та авторизації користувачів, керування майстрами, категоріями та послугами, а також механізми створення, редагування й обробки візитів. Розроблено функціонал автоматичного фінансового обліку з нарахуванням заробітної плати працівникам, веденням касових операцій та контролем фінансових транзакцій. Також створено аналітичний модуль для збору та візуалізації статистичних показників діяльності підприємства, зокрема доходів, кількості клієнтів, візитів і ефективності роботи персоналу. Окремо реалізовано модуль онлайн-запису, який забезпечує можливість самостійного бронювання послуг клієнтами, автоматичне формування доступних часових слотів, перевірку коректності даних та запобігання подвійним бронюванням. У результаті створено повноцінну CRM-систему, що автоматизує основні бізнес-процеси підприємств сфери послуг та підвищує ефективність управління клієнтською базою, персоналом і фінансами.

У висновках підведено підсумки виконаної роботи, визначено результати розробки та можливі напрями подальшого вдосконалення системи.

Кваліфікаційна робота викладена на 83 сторінках машинописного тексту, складається із вступу, 4 розділів, загальних висновків, переліку джерел посилання в кількості 30 джерел. Праця містить 4 таблиць та 32 рисунків.

Ключові слова: Діаграма, класи, фінанси, аналітика, CRM , CSS , PHP , SQL.

ABSTRACT

to the qualifying bachelor's thesis

CRM system for creating appointments

Student of 408 group: Hordiienko Oleksandr

Supervisor: PhD, (retired) Associate Professor

Antipova Kateryna Oleksandrivna

The relevance of the topic is determined by the need to automate customer management processes and organize the operation of beauty salons, which increases service efficiency and optimizes resource utilization.

The object of the study is the processes of planning and organizing the operation of a CRM system for appointment scheduling.

The subject of the study is the tools used for developing a CRM system.

The purpose of the study is to develop a system that ensures effective interaction between clients, administrators, and employees.

The qualification work consists of an introduction, __ sections, conclusions, and a list of references.

In the introduction, the relevance of the topic is justified, and the purpose, objectives, object, and subject of the study are defined, along with the practical significance of the obtained results.

In the first section, an analysis of the subject area is conducted, the features of system operation are considered, and existing CRM systems are reviewed, including their advantages and disadvantages. The main requirements for the future system are also defined, and the technical specification is formed.

The second section is devoted to the design of the CRM system. The system architecture is described, the choice of technologies is justified, and the database structure, class diagrams, and interaction diagrams are developed. The main system modules and their functionality are defined.

The third section presents the implementation of the software part of the system. The development process, user interface, and functional capabilities (customer

management, appointment scheduling, employee accounting, analytics) are described. System testing and efficiency evaluation are also carried out.

The fourth section presents the development and software implementation of a CRM system for managing appointments in service-oriented businesses are presented. During the development process, a relational database was designed using Microsoft SQL Server to ensure the storage and processing of information about users, branches, employees, clients, services, appointments, and financial transactions. The server-side component of the system was implemented using PHP and includes modules for user registration and authentication, management of employees, service categories and services, as well as mechanisms for creating, editing, and processing appointments. Automated financial accounting functionality was developed, including employee payroll calculation, cash register management, and financial transaction control. In addition, an analytical module was implemented to collect, process, and visualize key business performance indicators, such as revenue, number of clients, appointments, and staff efficiency. A separate online booking module was also developed, enabling clients to independently schedule services, automatically generate available time slots, validate booking data, and prevent double bookings. As a result, a fully functional CRM system was created that automates the core business processes of service enterprises and improves the efficiency of managing clients, personnel, and financial operations.

In the conclusions, the results of the work are summarized and directions for further development are outlined.The qualification work is presented on 83 pages of typewritten text, consists of an introduction, 4 sections, general conclusions, a list of references with 30 titles. The work contains 4 tables and 32 figures.

Keywords: Chart, classes, finance, analytics, CRM , CSS , PHP , SQL.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

API – Application Programming Interface (програмний інтерфейс застосунку)

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

CSS – Cascading Style Sheets (каскадні таблиці стилів)

HTML – HyperText Markup Language (мова розмітки гіпертексту)

JS – JavaScript (мова програмування для веб-розробки)

MVC – Model-View-Controller (архітектурний шаблон проєктування програмного забезпечення)

MySQL – система управління реляційними базами даних

PHP – Hypertext Preprocessor (серверна мова програмування)

REST API – Representational State Transfer Application Programming Interface

SaaS – Software as a Service (програмне забезпечення як сервіс)

SQL – Structured Query Language (мова структурованих запитів до баз даних)

UI – User Interface (користувацький інтерфейс)

UML – Unified Modeling Language (уніфікована мова моделювання)

ВСТУП

Актуальність теми кваліфікаційної роботи зумовлена стрімким розвитком сфери послуг та зростаючою необхідністю автоматизації процесів взаємодії з клієнтами. У сучасних умовах підприємства, що надають послуги за попереднім записом, стикаються з проблемами ефективного управління візитами, координації роботи персоналу та обробки великого обсягу клієнтських даних. Використання традиційних підходів до організації цих процесів часто призводить до втрати інформації, помилок у записах, дублювання даних та зниження якості обслуговування.

Впровадження CRM-систем дозволяє суттєво підвищити ефективність управління клієнтськими взаємовідносинами, автоматизувати процеси запису на послуги, забезпечити централізоване зберігання інформації та покращити контроль за виконанням візитів. Крім того, такі системи сприяють підвищенню продуктивності персоналу та прийняттю більш обґрунтованих управлінських рішень на основі аналітичних даних.

Науково-практичне значення роботи полягає у розробці підходів до проєктування та реалізації CRM-системи для управління візитами, що може бути використано як основа для створення аналогічних інформаційних систем у сфері послуг. Отримані результати можуть бути застосовані для автоматизації процесів у салонах краси, медичних закладах, сервісних центрах та інших організаціях, діяльність яких пов’язана з обслуговуванням клієнтів за записом.

Метою роботи є розробка CRM-системи для забезпечення ефективної взаємодії між клієнтами, адміністраторами та працівниками з метою автоматизації процесів управління візитами та підвищення ефективності роботи сервісної організації.

Відповідно до мети визначено такі завдання:

– проаналізувати предметну область та особливості процесів управління візитами;

– дослідити існуючі програмні рішення у сфері CRM-систем;

– визначити основні вимоги до програмної системи;

– спроєктувати структуру CRM-системи та її основні модулі;

– визначити структуру бази даних для зберігання інформації про клієнтів, послуги та візити;

– розробити та реалізуватим основні функціонали системи.

Об’єктом роботи є процеси планування та організації роботи crm системи для створення візитів.

Предметом роботи є інструменти розробки CRM-системи.

Обґрунтування необхідності розробки нової програмної системи базується на аналізі сучасного стану CRM-рішень. Існуючі системи, такі як Bitrix24, Kommo, Zoho CRM та Bookon CRM, забезпечують широкий функціонал для управління клієнтами, однак часто мають обмеження у гнучкості налаштувань, складності адаптації під специфічні бізнес-процеси або надлишкову функціональність, яка ускладнює використання у невеликих сервісних підприємствах. Це створює потребу у розробці більш спеціалізованого рішення, орієнтованого саме на управління візитами та роботою персоналу.

Основними напрямками проєктних рішень є використання клієнт-серверної архітектури, модульної структури системи та реляційної бази даних для забезпечення надійного зберігання та обробки інформації. Такий підхід дозволяє забезпечити масштабованість системи, її розширюваність та зручність подальшої підтримки.

Сфера застосування результатів роботи охоплює підприємства сфери послуг, зокрема салони краси, медичні центри, сервісні компанії та інші організації, які працюють за системою попереднього запису клієнтів. Розроблена система може бути використана як готове рішення або як основа для подальшого розвитку подібних інформаційних систем.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Об’єкт і предмет кваліфікаційної роботи

Об’єктом роботи є процеси планування та організації роботи CRM-системи створення візитів. Зазначені процеси охоплюють комплекс дій, пов’язаних із взаємодією між клієнтами, адміністраторами та працівниками сервісної організації, а також включають управління інформаційними потоками, що виникають під час запису, обслуговування та аналізу візитів.

У сучасних умовах розвитку сфери послуг спостерігається постійне зростання конкуренції між підприємствами, що зумовлює необхідність підвищення якості обслуговування клієнтів та оптимізації внутрішніх бізнес-процесів. Особливу роль у цьому відіграє ефективна організація роботи з клієнтами, яка включає своєчасний запис на послуги, коректне ведення бази даних, координацію роботи персоналу та контроль виконання замовлень. Використання застарілих або неавтоматизованих підходів призводить до втрати інформації, дублювання даних, помилок у записах та зниження загальної продуктивності підприємства.

Одним із ключових інструментів вирішення зазначених проблем є впровадження CRM-систем (Customer Relationship Management), які забезпечують автоматизацію процесів управління взаємодією з клієнтами. Такі системи дозволяють централізовано зберігати інформацію, автоматизувати запис на послуги, забезпечувати контроль виконання візитів, а також формувати аналітичні звіти для прийняття управлінських рішень.

Предметом роботи є інструменти розробки CRM-системи, зокрема методи, технології та програмні засоби, що використовуються для створення інформаційної системи управління візитами. До таких інструментів належать сучасні веб-технології, засоби розробки клієнт-серверних додатків, системи управління базами даних, а також підходи до проєктування програмного забезпечення.

Метою роботи є розробка CRM-системи, яка забезпечить ефективну взаємодію між клієнтами, адміністраторами та працівниками, а також дозволить автоматизувати основні процеси управління візитами. Реалізація такої системи

сприятиме підвищенню швидкості обробки інформації, зменшенню кількості помилок та покращенню якості обслуговування клієнтів.

Для досягнення поставленої мети необхідно виконати комплекс взаємопов’язаних задач, серед яких аналіз предметної області, дослідження існуючих програмних рішень, визначення функціональних та нефункціональних вимог до системи, проєктування структури програмного забезпечення, а також реалізація основного функціоналу для управління клієнтами, послугами та візитами.

1.2 Аналіз предметної області та функціональних особливостей

Предметна область даної кваліфікаційної роботи охоплює процеси планування та організації візитів клієнтів у сервісних організаціях. До таких організацій належать салони краси, медичні заклади, сервісні центри та інші підприємства, діяльність яких пов’язана з наданням послуг за попереднім записом. Основною особливістю цієї предметної області є необхідність ефективної координації взаємодії між клієнтами та персоналом, а також забезпечення безперервності та точності інформаційних потоків.

У межах предметної області виділяються основні учасники процесу, серед яких клієнти, адміністратори та працівники. Клієнти здійснюють запис на послуги, отримують обслуговування та можуть переглядати інформацію про свої візити. Адміністратори виконують функції координації, зокрема здійснюють реєстрацію клієнтів, формують записи на послуги, контролюють розклад та вирішують організаційні питання. Працівники (майстри) безпосередньо надають послуги та працюють відповідно до сформованого графіка.

Процес організації візитів включає декілька взаємопов’язаних етапів. На початковому етапі здійснюється формування бази клієнтів, яка містить персональні дані та історію взаємодії. Далі формується перелік послуг із зазначенням їх характеристик, тривалості та вартості. Після цього виконується планування графіку роботи працівників, що дозволяє визначити доступні часові інтервали для запису.

Наступним етапом є безпосередній запис клієнта на послугу, який може бути змінений або скасований у разі необхідності. Завершальним етапом є фіксація факту надання послуги та збереження інформації для подальшого аналізу.

Структурно система, що підтримує зазначені процеси, повинна складатися з взаємопов’язаних компонентів. До основних складових належать модуль управління клієнтами, модуль управління послугами, модуль управління візитами, модуль управління персоналом, а також модуль аналітики та звітності. Окремо виділяється модуль управління доступом, який забезпечує розмежування прав користувачів відповідно до їх ролей.

Функціональні особливості предметної області визначають вимоги до майбутньої системи. Серед них важливими є забезпечення швидкого доступу до актуальної інформації, підтримка одночасної роботи декількох користувачів, збереження цілісності та узгодженості даних, а також можливість масштабування системи. Крім того, система повинна забезпечувати зручний інтерфейс користувача, підтримувати пошук та фільтрацію даних, а також надавати засоби для формування аналітичної інформації.

Таблиця 1.1 - Аналіз системи, що розробляється

Розділ	Опис
Основні задачі системи	1. Аналізування та облік клієнтів, лідів і замовлень. 2. Управління послугами та їх категоріями. 3. Планування та облік візитів клієнтів до майстрів. 4. Генерація аналітичних звітів для керівництва. 5. Контроль доступу та прав користувачів. 6. Управління базою даних майстрів та працівників.
Користувачі системи	1. Глава філії / Компанія 2. Адміністратор 3. Майстер 4. Технічний адміністратор

Продовження Таблиці 1.1

Опис структури системи (схема)	Клієнт-серверна архітектура: [Користувачі] → [Веб-клієнт / Мобільний додаток] → [Сервер додатків] → [База даних] Користувачі – інтерфейси для всіх ролей	Сервер – бізнес-логіка, API, автоматизації База даних – зберігання даних про клієнтів, візити, послуги, майстрів, ролі
Сценарії роботи системи	Сценарій 1: Запис клієнта на послугу. Сценарій 2: Генерація аналітичного звіту. Сценарій 3: Керування правами користувачів.	
Основні таблиці БД	1. roles – ролі користувачів 2. visits – візити клієнтів 3. services – послуги	

	4. service_categories – категорії послуг 5. masters – майстри 6. users – користувачі системи
Засоби апаратної та програмної реалізації	Апаратні: сервер (фізичний або хмарний), ПК та мобільні пристрої користувачів. Програмні: сервер: PHP; база даних: MySQL; веб-клієнт: HTML/CSS/JS; хостинг, домен, SSL, резервне копіювання, аналітика

1.3 Огляд і аналіз існуючих програмних рішень

У сучасних умовах розвитку інформаційних технологій існує велика кількість програмних рішень, що забезпечують автоматизацію процесів управління взаємовідносинами з клієнтами. CRM-системи широко використовуються у сфері послуг для організації записів клієнтів, управління базами даних, планування візитів та аналізу діяльності підприємств різного масштабу від малих бізнесів до великих корпорацій.

Впровадження таких систем дозволяє значно підвищити ефективність роботи персоналу, зменшити кількість помилок, пов’язаних із людським фактором, а також забезпечити централізоване зберігання та обробку інформації. Крім того, сучасні CRM-рішення часто включають інструменти аналітики, автоматизації бізнес-процесів та інтеграції з іншими сервісами, що робить їх універсальними інструментами для управління клієнтською базою.

Різні програмні продукти відрізняються за своєю архітектурою, функціональними можливостями, рівнем масштабованості, способом розгортання та технологіями реалізації. Частина систем реалізується як хмарні рішення (SaaS), інші передбачають можливість локального встановлення на сервері підприємства, що дозволяє більш гнучко керувати даними та безпекою.

Тому для обґрунтування вимог до розроблюваної системи доцільно провести аналіз існуючих CRM-рішень, визначити їх основні функціональні можливості, переваги та недоліки, а також оцінити доцільність використання певних архітектурних та технологічних підходів у рамках даної кваліфікаційної роботи. Отримані результати аналізу дозволять сформувати обґрунтовані вимоги до майбутньої системи та визначити напрямки її реалізації.

Таблиця 1.2 - Аналогічні систем

Параметр	Опис
Назва	Bitrix24 CRM
Виробник	Bitrix, Inc.
Архітектура	SaaS та on-premise; клієнт-сервер; розширення через REST API
Мова реалізації	PHP (Bitrix Framework) для on-premise ядра
Основні функції / характеристики	Ліди, угоди, воронки продажів; телефонія; автоматизація бізнес-процесів; маркетинг та розсилки; завдання і проєкти; аналітика продажів; інтеграції
Переваги	Єдина платформа «все в одному»; потужна екосистема; локальне розгортання; гнучкий REST API; масштабованість
Недоліки	Складність для розробників; крута крива навчання

Таблиця 1.3 – Аналогічні CRM-системи Kommo та Zoho CRM

Параметр	Kommo (раніше amoCRM)	Zoho CRM
Назва	Kommo (раніше amoCRM)	Zoho CRM
Виробник	Kommo	Zoho Corporation
Архітектура	Хмарна SaaS CRM, веб-клієнт та мобільні додатки	Хмарна багатокористувацька платформа; інтеграція в екосистему Zoho
Мова реалізації	Не розкривається публічно	Не розкривається публічно
Основні функції / характеристики	Месенджер-центрична робота; спільна «Вхідна»; воронки продажів; автоматизації та тригери; VoIP/телефонія; веб-форми; мобільні застосунки; відкритий API	Ведення лідів і контактів; конструктор Canvas; автоматизації; AI-помічник Zia; аналітика та звіти; сотні інтеграцій
Переваги	Сильні інтеграції з месенджерами; інтуїтивний інтерфейс; швидке хмарне	Велика гнучкість; приваблива ціна; інтеграція з Zoho-

	розгортання; автоматизації з коробки; підтримка	додатками; широкий каталог функцій
Недоліки	Обмежена кастомізація; поодинокі баги; крива навчання для деяких функцій; висока ціна для великих команд	Тільки хмара; ускладнення при великій кількості налаштувань

Таблиця 1.4 – Аналогічна CRM-система Bookon CRM

Параметр	Опис
Назва	Bookon CRM
Виробник	Binotel
Архітектура	Хмарна (SaaS) та локальна установка; веб-клієнт, мобільні додатки
Мова реалізації	Не розкривається публічно (ймовірно PHP + JS)
Основні функції / характеристики	Управління контактами та лідами; воронки продажів; телефонія (IP/VoIP); автоматизації бізнес-процесів; аналітика та звіти; інтеграції з месенджерами; мобільні додатки
Переваги	Інтегрована телефонія; простий веб-інтерфейс; швидке розгортання; підтримка українською мовою; масштабованість для малого та середнього бізнесу
Недоліки	Обмежені можливості кастомізації; відсутність глибокого API для розширеної інтеграції; частина функцій доступна лише у платній версії

Серед найбільш поширених CRM-рішень, що використовуються на сучасному ринку, можна виділити такі програмні продукти як Bitrix24 CRM, Kommo (раніше amoCRM), Zoho CRM та Bookon CRM. Дані системи відрізняються між собою за підходами до організації бізнес-процесів, архітектурними рішеннями та рівнем функціональності, однак усі вони спрямовані на автоматизацію взаємодії з клієнтами та підвищення ефективності роботи підприємств.

CRM-система Bitrix24 є комплексним рішенням, яке поєднує в собі інструменти управління продажами, комунікаціями та внутрішніми бізнес-процесами. Вона підтримує як хмарне розгортання, так і локальну інсталяцію, що дозволяє використовувати її як у малих компаніях, так і у великих організаціях. Завдяки широкому набору функцій система забезпечує повний цикл управління клієнтами, включаючи роботу з лідами, угодами, завданнями та аналітикою.

CRM-система Kommo (раніше amoCRM) приклад зображений на рисунку 1.1 орієнтована переважно на автоматизацію продажів та роботу з клієнтами через месенджери та цифрові канали комунікації. Основний акцент у даній системі зроблено на швидкість обробки заявок та зручність взаємодії з клієнтами через єдину вхідну скриньку повідомлень. Це робить її зручною для бізнесів, які активно використовують онлайн-комунікації.

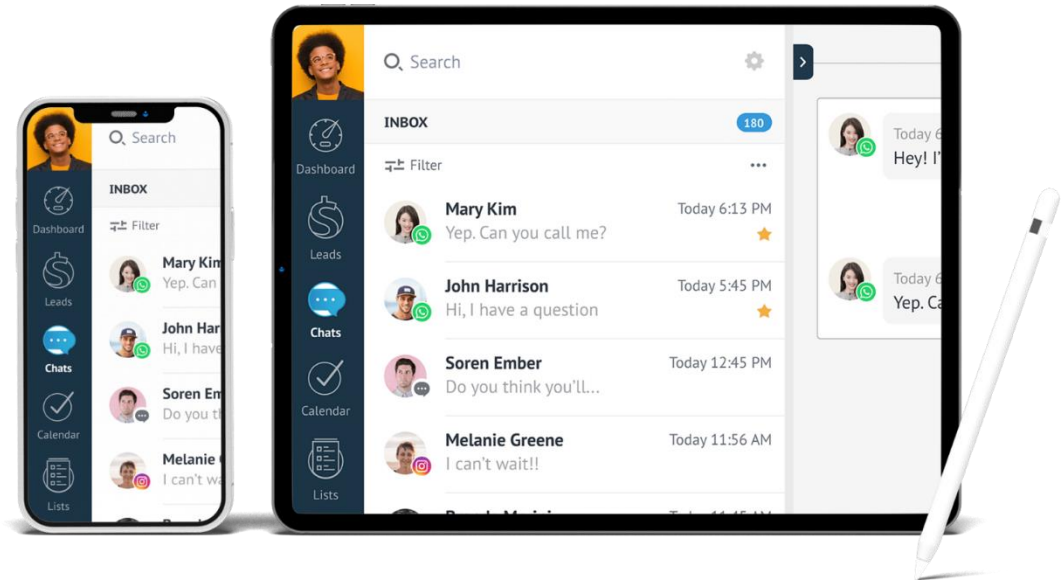


Рисунок 1.1 – Kommo CRM

Zoho CRM на рисунку 1.2 є гнучкою хмарною платформою, що входить до складу великої екосистеми Zoho. Вона надає широкі можливості для налаштування бізнес-процесів, використання аналітики та штучного інтелекту, а також інтеграції з іншими сервісами. Система орієнтована на компанії, які потребують масштабованого та універсального рішення для управління клієнтською базою.

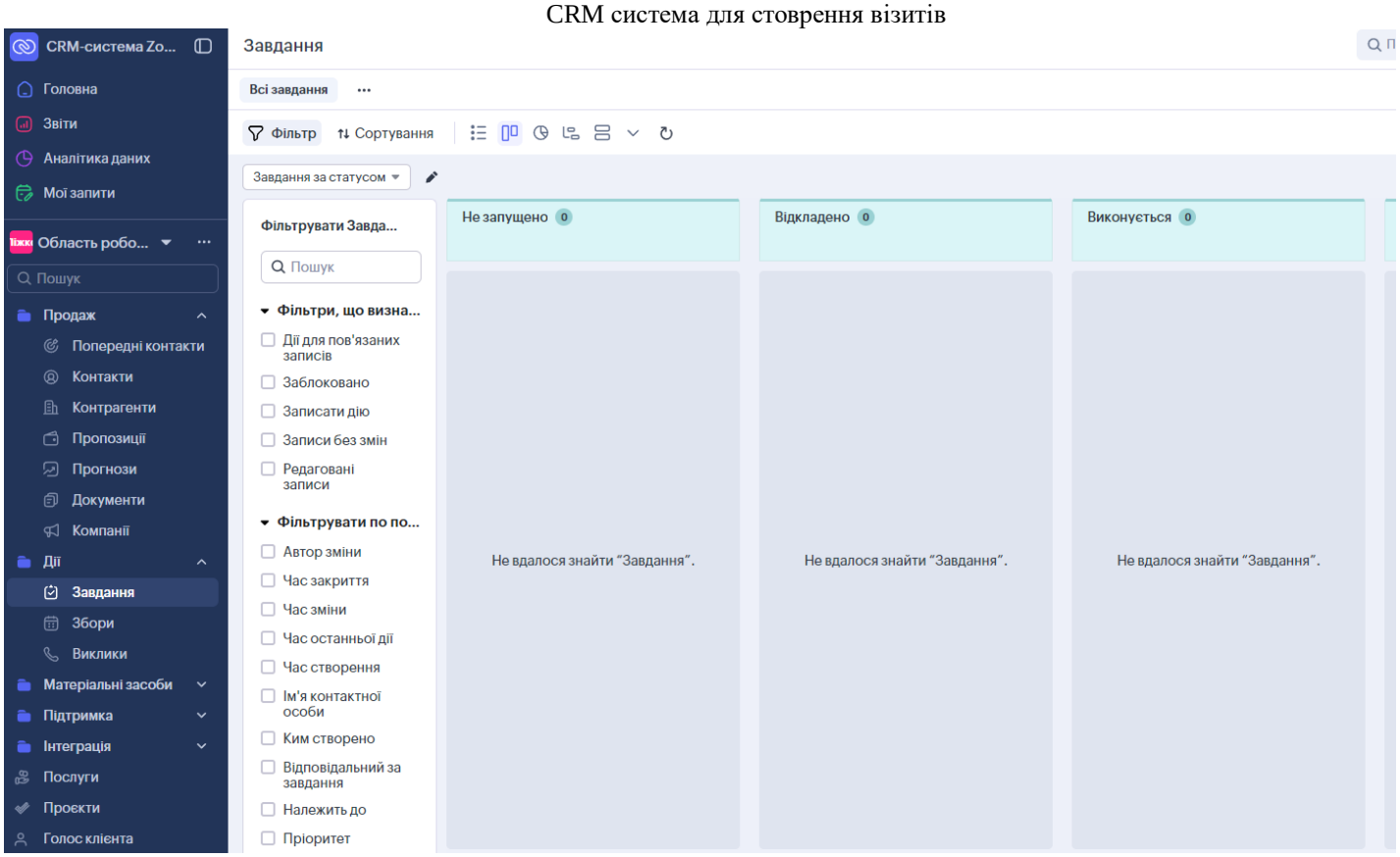


Рисунок 1.2 - Zoho CRM

Окремо варто виділити CRM-систему Bookon CRM на рисунку 1.3, яка орієнтована на малий та середній бізнес і поєднує у собі функції управління контактами, телефонії та базової аналітики. Вона забезпечує відносно просту інтеграцію з бізнес-процесами та зручний інтерфейс, однак має обмеження у частині розширення функціоналу та глибокої кастомізації.

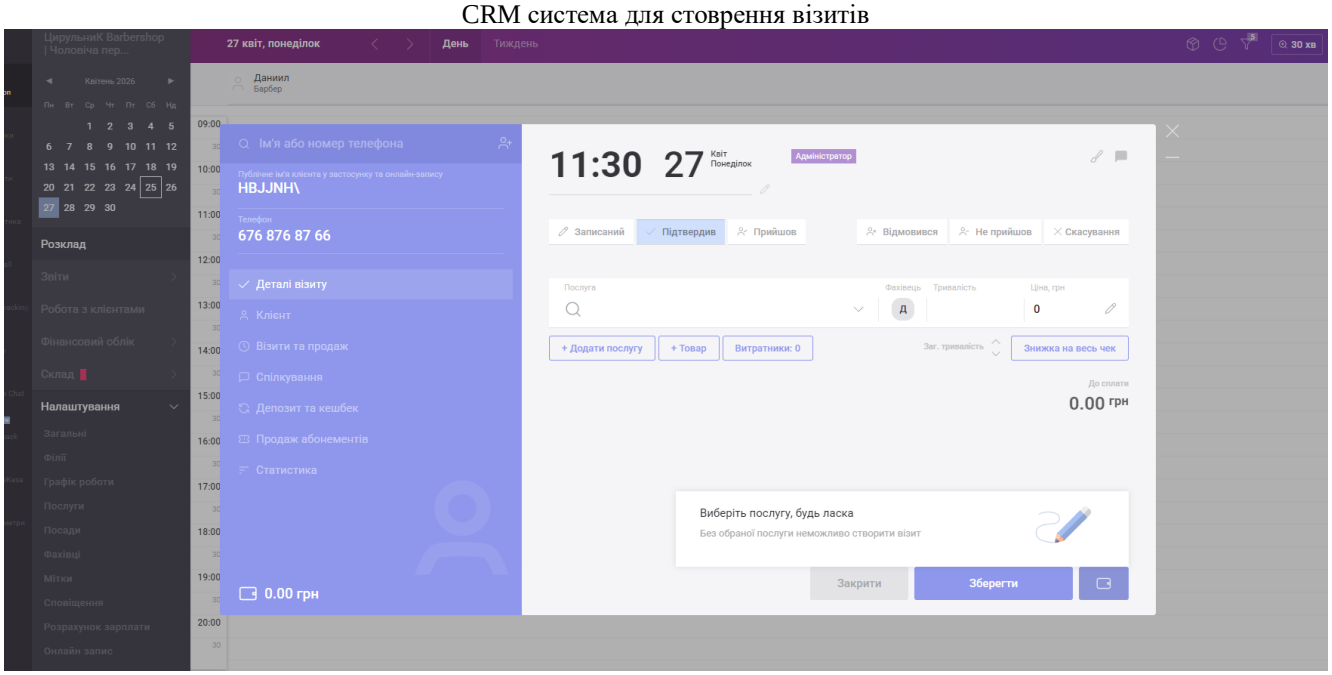


Рисунок 1.3 - Bookon CRM

Таким чином, розглянуті CRM-системи демонструють різні підходи до реалізації автоматизації бізнес-процесів. Одні з них орієнтовані на комплексне управління підприємством, інші на вузькоспеціалізовані задачі, такі як обробка лідів або комунікація з клієнтами. Це дозволяє визначити основні тенденції розвитку CRM-рішень та сформувати вимоги до розроблюваної системи.

Висновки до розділу 1

У першому розділі кваліфікаційної роботи було виконано комплексний аналіз предметної області, що охоплює процеси планування та організації роботи CRM-системи для створення та управління візитами клієнтів у сервісних організаціях. Розглянуто специфіку діяльності підприємств сфери послуг, де ключове значення має ефективна взаємодія між клієнтами, адміністраторами та виконавцями послуг, а також своєчасна та точна організація процесів запису на обслуговування.

У ході аналізу встановлено, що предметна область характеризується високою залежністю від точності обробки інформації, швидкості доступу до даних та необхідності координації великої кількості взаємопов'язаних процесів. До основних з них належать ведення клієнтської бази, формування та управління

переліком послуг, планування робочого графіку працівників, організація записів на візити та контроль їх виконання. Всі ці процеси мають бути узгоджені між собою для забезпечення стабільної роботи сервісної організації.

Було визначено, що в умовах зростання обсягів інформації та збільшення кількості клієнтів традиційні методи обліку та планування стають неефективними. Ручне ведення записів або використання простих інструментів призводить до виникнення помилок, дублювання даних, втрати інформації та зниження якості обслуговування. Це обґрунтовує необхідність впровадження автоматизованих CRM-систем, які дозволяють централізувати дані та оптимізувати бізнес-процеси.

Окрему увагу в розділі приділено аналізу існуючих програмних рішень у сфері CRM. Було розглянуто такі системи, як Bitrix24 CRM, Kommo (раніше amoCRM), Zoho CRM та Bookon CRM. Проведений аналіз показав, що кожна з систем має свої особливості, переваги та обмеження. Зокрема, Bitrix24 характеризується широким функціоналом і можливістю комплексної автоматизації бізнесу, але має складний інтерфейс та високу складність освоєння. Kommo орієнтована на комунікацію через месенджери та швидку обробку лідів, однак має обмеження у гнучкості налаштувань. Zoho CRM є потужною хмарною платформою з великою кількістю інтеграцій, але може бути складною у конфігурації. Bookon CRM орієнтована на малий та середній бізнес, забезпечує базовий функціонал, проте має обмеження у розширенні можливостей.

Також було проаналізовано загальні підходи до побудови CRM-систем, включаючи використання клієнт-серверної архітектури, хмарних технологій та модульного підходу до організації функціоналу. Визначено, що найбільш ефективними є системи, які поєднують гнучкість налаштувань, масштабованість та можливість інтеграції з іншими сервісами.

На основі проведеного аналізу сформовано загальні вимоги до розроблюваної системи. Вона повинна забезпечувати автоматизацію процесів управління клієнтами та візитами, підтримувати багаторівневу систему доступу,

дозволяти ефективне планування роботи персоналу, а також надавати інструменти для аналітики та контролю діяльності.

2 МОДЕЛЮВАННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА РОЗРОБКА ВИМОГ ДО CRM-СИСТЕМИ

2.1 Аналіз сучасного стану інструментарію, моделей та методів розробки CRM-систем

У сучасних умовах розвитку інформаційних технологій створення CRM-систем базується на використанні широкого спектра інструментів, моделей та методів, що забезпечують ефективну автоматизацію процесів управління взаємовідносинами з клієнтами. Згідно з сучасними дослідженнями у сфері програмної інженерії та інформаційних систем, ключову роль відіграє комплексний підхід, який поєднує аналіз бізнес-процесів, програмну інженерію та сучасні технології обробки даних [1, 2].

Сучасні CRM-системи розглядаються не лише як інструменти зберігання інформації про клієнтів, а як комплексні інформаційні платформи для підтримки прийняття управлінських рішень. Вони охоплюють повний цикл взаємодії з клієнтом: від первинного контакту до формування аналітичної звітності та прогнозування подальшої взаємодії.

Одним із основних напрямків розвитку CRM-систем є використання сучасних архітектурних підходів. Зокрема, традиційні монолітні системи поступово замінюються мікросервісною архітектурою, яка забезпечує масштабованість, відмовостійкість та гнучкість системи [3,4]. Використання мікросервісів дозволяє розділити систему на незалежні модулі, кожен з яких відповідає за окрему функціональність, що значно спрощує процес розробки, тестування та подальшого супроводу програмного забезпечення.

Окрему увагу в сучасних дослідженнях приділяють питанням інтеграції CRM-систем із зовнішніми сервісами та API. Це дозволяє забезпечити обмін даними між різними інформаційними системами підприємства, що підвищує загальний рівень автоматизації бізнес-процесів та зменшує кількість ручних операцій.

Важливе значення мають методи моделювання предметної області. У сучасних дослідженнях активно застосовуються UML-діаграми, методи структурного аналізу та об’єктно-орієнтованого проєктування, що дозволяють формалізувати бізнес-процеси та визначити взаємозв’язки між компонентами системи [1,2,5]. Використання таких підходів забезпечує більш точне проєктування програмного забезпечення та зменшує ймовірність виникнення помилок на етапі реалізації.

Додатково застосовуються CASE-засоби (Computer-Aided Software Engineering), які дозволяють автоматизувати процес побудови моделей системи, включаючи діаграми класів, прецедентів та потоків даних. Це значно підвищує ефективність етапу проєктування та забезпечує кращу узгодженість між вимогами та реалізацією.

Окрему роль відіграють методи аналітичної обробки даних у CRM-системах. Використання підходів до аналізу даних та моделей машинного навчання дозволяє здійснювати аналіз поведінки клієнтів, прогнозування попиту та персоналізацію послуг [9,10]. Застосування таких моделей підвищує ефективність взаємодії з клієнтами, дозволяє формувати індивідуальні пропозиції та покращує якість управлінських рішень.

Крім того, сучасні CRM-системи інтегрують технології обробки великих обсягів даних (Big Data), що дає можливість аналізувати значні масиви інформації про клієнтів у реальному часі. Це особливо важливо для підприємств сфери послуг, де швидкість прийняття рішень безпосередньо впливає на якість обслуговування.

Ще одним важливим напрямком є використання хмарних технологій. Хмарні платформи забезпечують доступність системи з будь-якого пристрою, масштабованість ресурсів та інтеграцію з іншими інформаційними системами підприємства [3]. Завдяки цьому CRM-системи стають більш гнучкими та економічно ефективними у впровадженні.

Також у наукових роботах підкреслюється необхідність реінжинірингу бізнес-процесів перед впровадженням CRM-систем. Автоматизація неефективних

процесів без їх попередньої оптимізації може призвести до зниження загальної ефективності системи, тому важливим етапом є детальний аналіз, моделювання та вдосконалення бізнес-процесів відповідно до цифрового середовища.

Таким чином, аналіз сучасного стану інструментарію, моделей та методів розробки CRM-систем показує, що ефективна система повинна базуватися на поєднанні сучасних архітектурних підходів, методів моделювання, аналітичних інструментів, технологій обробки даних та принципів автоматизації бізнес-процесів [1, 2, 3, 4, 8, 9, 10]. Лише комплексне використання зазначених підходів дозволяє створити гнучку, масштабовану та ефективну CRM-систему, яка відповідає сучасним вимогам цифрової трансформації підприємств.

2.2 Моделювання предметної області та специфікація вимог до програмного забезпечення

Моделювання предметної області є важливим етапом розробки програмного забезпечення, оскільки дозволяє формалізувати основні бізнес-процеси, визначити структуру системи та встановити взаємозв’язки між її компонентами. У рамках даної кваліфікаційної роботи моделювання здійснюється для процесів управління клієнтами, послугами та візитами у CRM-системі.

Для опису предметної області використовуються сучасні підходи структурного та об’єктно-орієнтованого аналізу. Зокрема, застосовуються UML-діаграми, які дозволяють відобразити основні аспекти функціонування системи. До основних моделей належать:

– діаграма варіантів використання (Use Case), яка відображає взаємодію користувачів із системою;

– діаграма класів, що описує структуру даних та взаємозв’язки між сутностями;

– діаграма послідовностей, яка демонструє порядок виконання операцій у системі;

– діаграма діяльності, що відображає бізнес-процеси та потоки виконання дій.

Основними акторами системи є клієнт, адміністратор та працівник (майстер). Клієнт здійснює запис на послугу, переглядає інформацію про візити та отримує обслуговування. Адміністратор відповідає за управління записами, координацію роботи персоналу та ведення бази даних. Працівник виконує послуги відповідно до сформованого графіка.

У межах моделювання визначаються основні сутності системи, серед яких клієнти, послуги, категорії послуг, візити, користувачі та ролі. Для кожної сутності визначаються атрибути та зв’язки, що дозволяє сформувати логічну модель бази даних.

Для графічного представлення моделей доцільно використовувати CASE-засоби, такі як StarUML, Draw.io або інші інструменти, що підтримують побудову UML-діаграм. Це забезпечує наочність моделей та спрощує процес їх аналізу.

Важливим етапом є формування специфікації вимог до програмного забезпечення (SRS), яка визначає функціональні та нефункціональні вимоги до системи. Специфікація вимог є основою для подальшої розробки, тестування та впровадження програмного продукту.

Функціональні вимоги описують основні можливості системи. До них належать:

– реєстрація та авторизація користувачів;

– управління клієнтами та їх даними;

– управління послугами та категоріями;

– створення, редагування та скасування візитів;

– формування звітів та аналітичної інформації;

– управління правами доступу.

Нефункціональні вимоги визначають характеристики якості системи. До них відносяться:

– продуктивність (швидка обробка запитів);

– надійність (збереження та відновлення даних);

– безпека (захист даних користувачів);

– масштабованість (можливість розширення системи);

– зручність використання (інтуїтивний інтерфейс).

Таким чином, проведене моделювання предметної області дозволяє сформувати цілісне уявлення про структуру та функціонування системи, а також визначити вимоги до програмного забезпечення. Це створює основу для подальшого проєктування та реалізації CRM-системи управління візитами.

1) Призначення та межі проєкту:

1.1) призначення системи (застосунку), для якої розробляється програмне забезпечення;

Вебзастосунок CRM-системи призначений для автоматизації процесів управління клієнтами, записами на візити та обліком послуг у сервісній організації. Система забезпечує взаємодію між клієнтами, адміністраторами та майстрами, а також дозволяє вести базу клієнтів, розклад записів і історію наданих послуг.

1.2) погодження, що ухвалені в програмній документації;

– система реалізується як вебзастосунок;

– технології: HTML, CSS, JavaScript, PHP;

– база даних: MySQL;

– середовище розробки та запуску: OpenServer;

– архітектура: клієнт-серверна;

– доступ до системи здійснюється через веббраузер.

1.3) межі проєкту ПЗ;

– розробка охоплює вебінтерфейс та серверну логіку CRM;

– включає модулі: користувачі, клієнти, послуги, майстри, візити;

– не передбачено мобільний застосунок;

– не реалізується інтеграція з платіжними системами;

– система працює у локальному або серверному середовищі.

2) Загальний опис:

2.1) сфера застосування;

Система застосовується у сфері надання послуг (наприклад, салони краси, сервісні центри) для автоматизації записів клієнтів, управління розкладом майстрів та обліку наданих послуг.

2.2) характеристики користувачів;

– адміністратор – керування системою, користувачами, записами та послугами;

– майстер – перегляд розкладу та виконаних візитів;

– клієнт – запис на послуги та перегляд власних візитів;

– користувачі мають базові навички роботи з вебсистемами.

2.3) загальна структура і склад системи;

– клієнтська частина (HTML, CSS, JS);

– серверна частина (PHP);

– база даних (MySQL);

– модулі: авторизація, користувачі, клієнти, послуги, візити, аналітика.

2.4) загальні обмеження;

– робота системи залежить від наявності браузера та інтернет/локального сервера;

– обмеження продуктивності залежать від конфігурації сервера OpenServer;

– одночасна кількість користувачів залежить від ресурсів БД і сервера.

3) Функції системи:

3.1) управління користувачами:

3.1.1) опис функції;

Функція забезпечує створення, редагування та видалення користувачів системи, а також призначення ролей (адміністратор, майстер, клієнт).

3.1.2) вхідна і вихідна інформація;

– вхідна: дані користувача (логін, пароль, роль);

– вихідна: запис у базі даних, статус операції.

3.1.3) функціональні вимоги;

– перевірка унікальності логіну;

– збереження паролів у захищеному вигляді;

– розмежування доступу за ролями.

3.2) управління клієнтами:

3.2.1) опис функції;

Функція забезпечує ведення бази клієнтів, включаючи додавання та редагування інформації.

3.2.2) вхідна і вихідна інформація;

– вхідна: дані клієнта (ПІБ, телефон, email);

– вихідна: запис у БД, список клієнтів.

3.2.3) функціональні вимоги;

– пошук клієнтів;

– редагування даних;

– перевірка коректності введення.

3.3) управління записами (візитами):

3.3.1) опис функції;

Функція дозволяє створювати записи клієнтів на послуги до майстрів.

3.3.2) вхідна і вихідна інформація;

– вхідна: клієнт, майстер, послуга, дата, час;

– вихідна: запис у календарі, підтвердження.

3.3.3) функціональні вимоги;

– перевірка зайнятості часу;

– уникнення конфліктів записів;

– можливість скасування візиту.

4) Вимоги до інформаційного забезпечення:

4.1) джерела і зміст вхідної інформації (даних);

– дані користувачів системи;

– інформація про клієнтів;

– дані про послуги та майстрів;

– записи про візити.

4.2) нормативно-довідкова інформація;

– довідник ролей користувачів;

– довідник послуг;

– довідник статусів записів.

4.3) вимоги до способів організації, збереження та ведення інформації;

– зберігання даних у MySQL;

– використання реляційної структури БД;

– регулярна перевірка цілісності даних.

5) Вимоги до технічного забезпечення.

– персональний комп’ютер або сервер;

– підтримка OpenServer;

– мережеве підключення для вебдоступу.

6) Вимоги до програмного забезпечення:

6.1) архітектура програмної системи;

Клієнт-серверна архітектура.

6.2) системне програмне забезпечення;

Windows / Linux + OpenServer.

6.3) мережне програмне забезпечення;

HTTP/HTTPS протокол.

6.4) програмне забезпечення ведення інформаційної бази;

MySQL.

6.5) мова і технологія розробки ПЗ;

HTML, CSS, JavaScript, PHP.

7) Вимоги до зовнішніх інтерфейсів:

7.1) інтерфейс користувача;

Вебінтерфейс з формами, таблицями та меню навігації.

7.2) апаратний інтерфейс;

Стандартний ПК або ноутбук.

7.3) програмний інтерфейс;

PHP-скрипти та MySQL-запити.

7.4) комунікаційний протокол;

HTTP/HTTPS.

8) Властивості програмного забезпечення:

8.1) доступність;

Система доступна через браузер 24/7.

8.2) супроводжуваність;

Модульна структура коду.

8.3) переносимість;

Може працювати на різних ОС.

8.4) продуктивність;

Швидка обробка запитів при середньому навантаженні.

8.5) надійність;

Захист від втрати даних при роботі з БД.

8.6) безпека;

Базова авторизація та контроль доступу.

9) Інші вимоги.

– резервне копіювання бази даних;

– захист від SQL-ін’єкцій;

– можливість розширення функціоналу системи.

Висновки до розділу 2

У другому розділі кваліфікаційної роботи виконано комплексне дослідження сучасного стану інструментарію, моделей та методів розробки CRM-систем, а також здійснено детальне моделювання предметної області та сформовано специфікацію вимог до програмного забезпечення.

Проведений аналіз наукових джерел та сучасних підходів до розробки інформаційних систем показав, що CRM-системи на сьогоднішній день є складними багатокомпонентними програмними продуктами, які поєднують у собі функції управління клієнтськими даними, автоматизації бізнес-процесів та

аналітичної обробки інформації. Встановлено, що найбільш перспективними підходами до їх реалізації є використання мікросервісної архітектури, принципів об’єктно-орієнтованого проєктування, а також сучасних засобів моделювання, зокрема UML-діаграм. Застосування зазначених методів дозволяє підвищити масштабованість, гнучкість та адаптивність програмних рішень до змін бізнес-вимог.

У процесі моделювання предметної області було визначено основні сутності системи, їх характеристики та взаємозв’язки, а також описано ключові бізнес-процеси, пов’язані з управлінням клієнтами, послугами та візитами. Результати моделювання дозволили формалізувати структуру майбутньої CRM-системи та визначити логіку взаємодії між її компонентами та користувачами різних ролей, що забезпечує цілісне розуміння функціонування програмного продукту ще на етапі проєктування.

Окрему увагу приділено формуванню специфікації вимог до програмного забезпечення (SRS), яка охоплює функціональні та нефункціональні вимоги, вимоги до інформаційного забезпечення, технічної інфраструктури, програмної реалізації та зовнішніх інтерфейсів. Чітко визначені вимоги створюють основу для подальшої реалізації системи та забезпечують відповідність розробки очікуванням кінцевих користувачів і сучасним стандартам створення програмного забезпечення.

Виконані в межах розділу дослідження дозволили сформувати теоретичну базу та структурне уявлення про майбутню CRM-систему, що є необхідною передумовою для переходу до етапу її проєктування та практичної реалізації. Отримані результати забезпечують узгодженість між вимогами, моделями та майбутньою архітектурою системи, що підвищує якість та обґрунтованість подальших етапів розробки програмного продукту.

3 ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ CRM-СИСТЕМИ

3.1 Проєктування взаємодії користувачів із системою

У цьому підрозділі здійснюється проєктування поведінкової моделі CRM-системи із застосуванням UML-діаграм прецедентів рисунок 3.1 та діаграм взаємодії. Метою даного етапу є формалізація логіки функціонування системи, визначення ролей користувачів, їхніх прав доступу та опис основних сценаріїв взаємодії з функціональними модулями програмного забезпечення.

Розроблювана CRM-система побудована на принципах розмежування доступу та ролей, що забезпечує структуровану організацію роботи користувачів та підвищує рівень безпеки обробки даних. У системі визначено три основні категорії користувачів: сервіс-адміністратор, головна філія та майстер. Кожна з ролей характеризується окремим набором функціональних можливостей відповідно до її бізнес-функції.

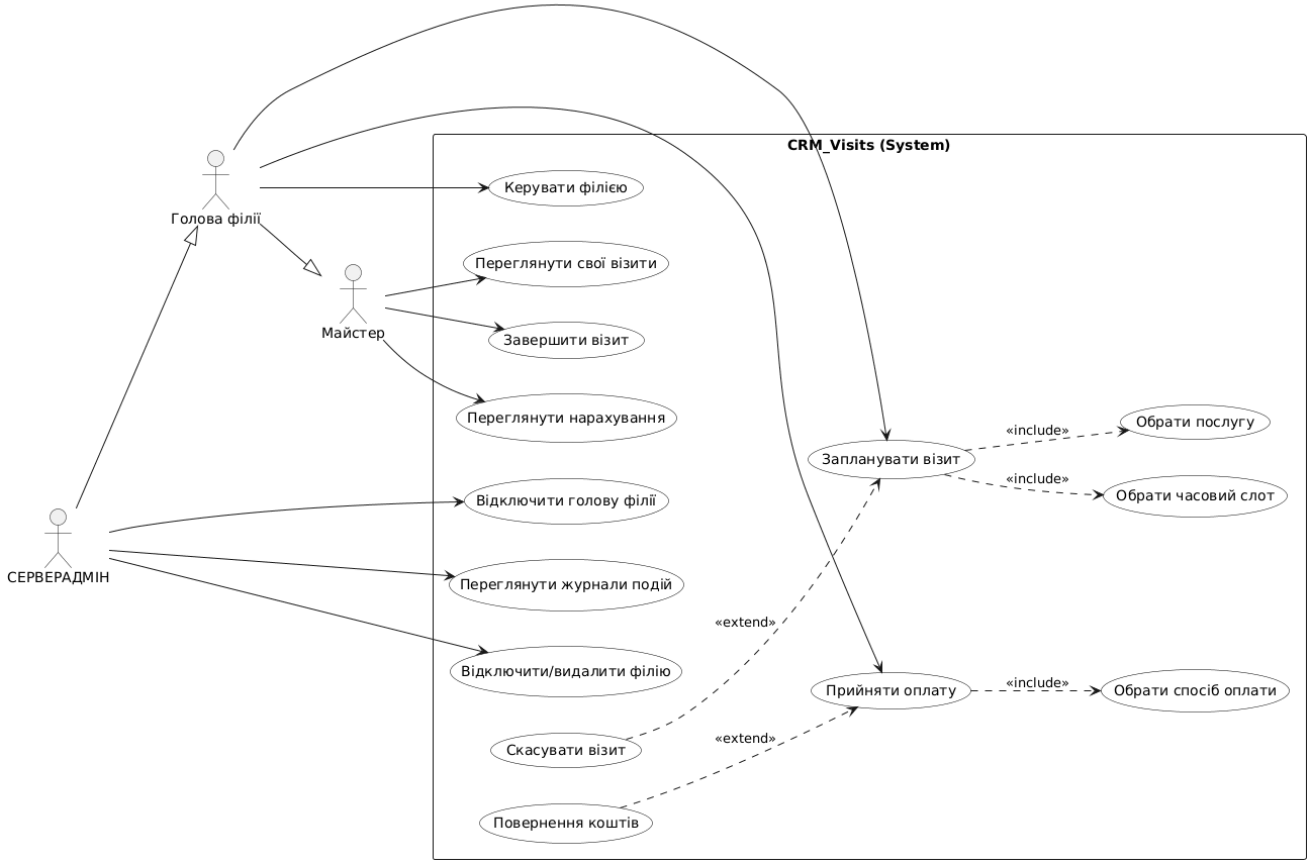


Рисунок 3.1 - Діаграма прецедентів

Сервіс-адміністратор виконує роль центрального керуючого вузла системи. До його основних функцій належить адміністрування користувачів, керування структурою філій, контроль доступу та моніторинг системних подій. Крім того, адміністратор має можливість перегляду журналів дій, що дозволяє забезпечувати контроль цілісності та коректності роботи системи в цілому.

Головна філія відповідає за організаційно-управлінський рівень системи та забезпечує координацію роботи підпорядкованих підрозділів. У межах своєї ролі користувач може здійснювати створення та редагування філій, керувати кадровим складом, а також контролювати виконання послуг та ефективність роботи майстрів. Таким чином, даний рівень виступає проміжною ланкою між адміністративним управлінням і виконавцями.

Майстер є виконавцем бізнес-процесів системи та безпосередньо взаємодіє з клієнтами. Його функціональність включає перегляд індивідуального розкладу, доступ до списку призначених візитів, виконання послуг та підтвердження їх завершення. Дана роль є ключовою з точки зору реалізації основної бізнес-логіки системи.

Особливу увагу в рамках проєктування приділено процесу управління візитами клієнтів, який є центральним бізнес-процесом CRM-системи. Він включає послідовність дій: вибір послуги, визначення доступного часу, призначення майстра, створення запису та подальше виконання візиту. У процесі також враховується перевірка доступності ресурсів, що дозволяє уникати конфліктів у розкладі та забезпечує коректність планування.

Описана модель взаємодії користувачів із системою є основою для подальшого проєктування діаграм послідовності та реалізації програмної частини CRM-системи.

Для детального аналізу функціонування розроблюваної інформаційної системи необхідно вийти за межі статичного опису структури та зосередитися на динамічних аспектах її роботи. Це досягається шляхом побудови діаграм послідовності та комунікацій, які описують життєвий цикл запитів від моменту

ініціації користувачем до фіксації змін у сховищі даних. У межах даного проекту було деталізовано шість ключових сценаріїв, що становлять функціональне ядро CRM-системи для салонів краси та центрів надання послуг.

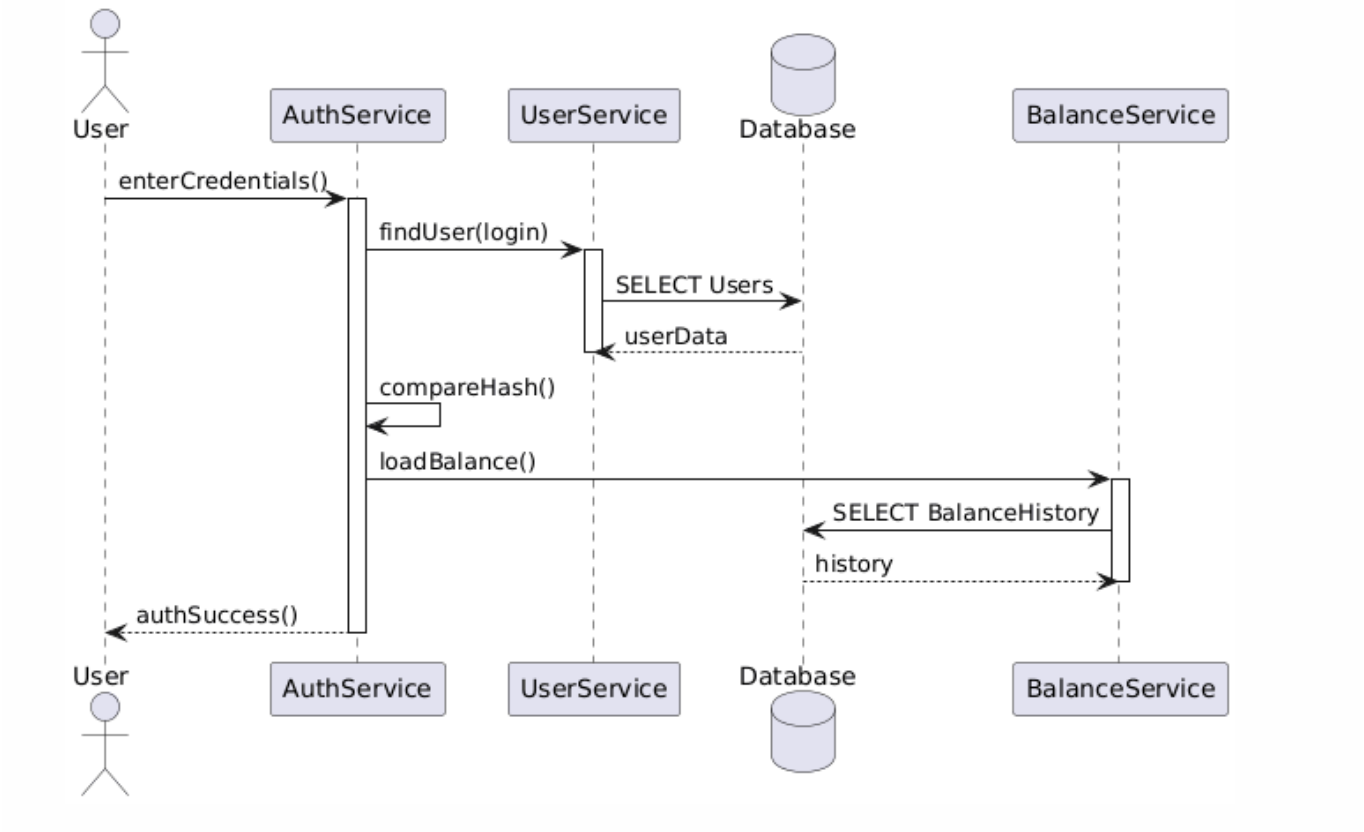


Рисунок 3.2 - Авторизація користувача (Sequence)

Першочерговим етапом будь-якої сесії є процес автентифікації користувача. На відповідній діаграмі рисунок 3.2 послідовності відображено складний механізм перевірки прав доступу, де сервіс авторизації AuthService виконує роль координатора. Після отримання облікових даних він звертається до сервісу користувачів для ідентифікації об’єкта в базі даних. Важливою особливістю реалізації є те, що порівняння хешованих значень паролів відбувається на рівні бізнес-логіки сервера, що запобігає витоку конфіденційної інформації. Після успішної перевірки система автоматично ініціює завантаження фінансових показників через BalanceService, що дозволяє користувачу отримати актуальний стан рахунків відразу після входу в інтерфейс.

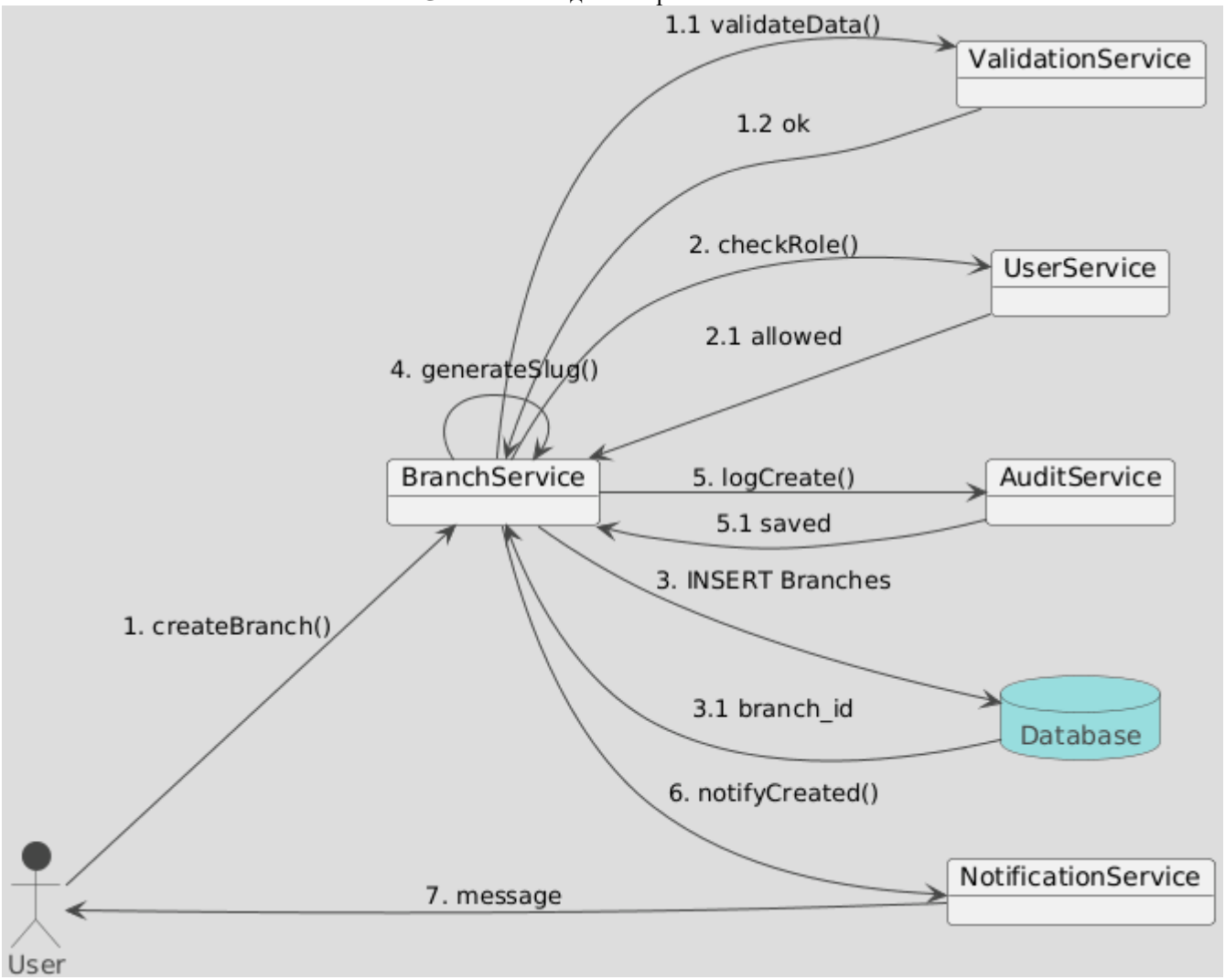


Рисунок 3.3 – Створення філіала

Процес управління організаційною структурою, зокрема створення нової філії, реалізовано через комунікаційну діаграму рисунок 3.3, яка демонструє принцип слабкої зв’язності компонентів. Сервіс філій BranchService не просто записує дані в таблицю, а попередньо валідує вхідні параметри та перевіряє рольову модель користувача. Система автоматично генерує унікальні ідентифікатори для веб-адрес та забезпечує цілісність даних через механізм аудиту, де кожен крок логується в окремому сервісі AuditService. Завершується сценарій асинхронним сповіщенням через NotificationService, що підтверджує успішність операції на рівні всієї інфраструктури.

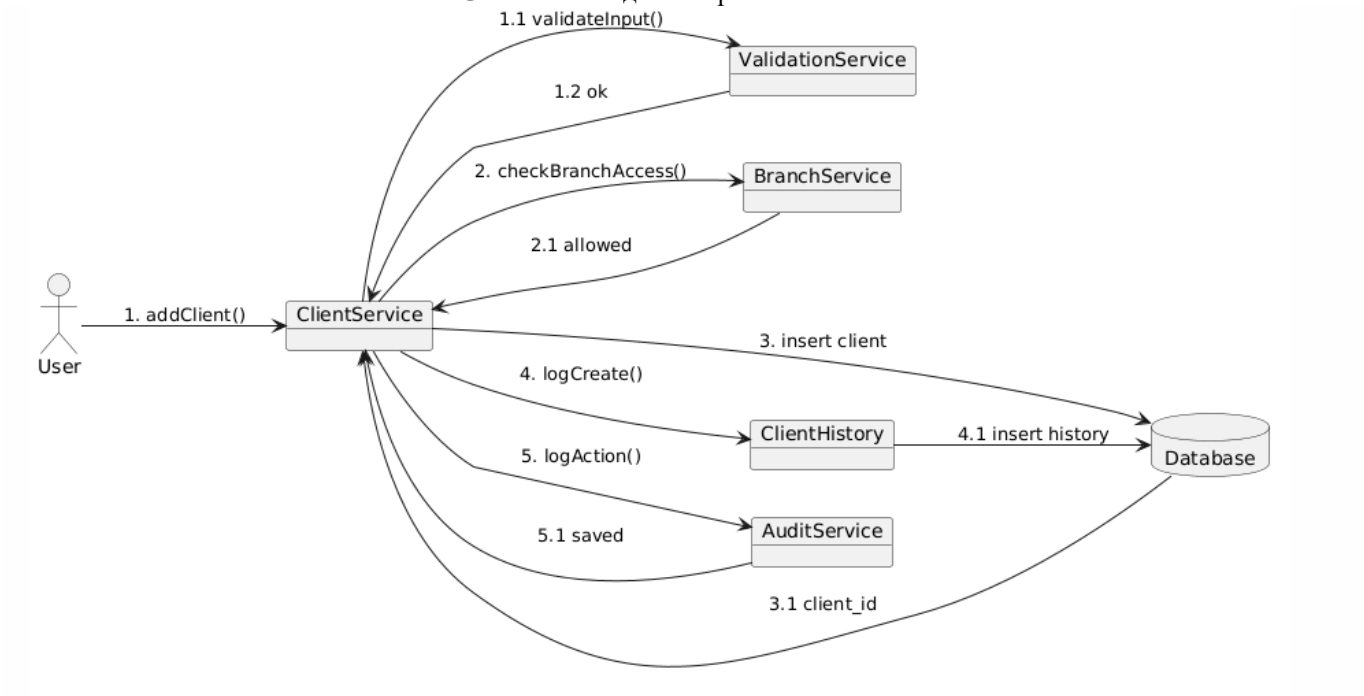


Рисунок 3.4 - Створення клієнта

Аналогічний підхід застосовано і при реєстрації нових клієнтів рисунок 3.4. Тут особлива увага приділяється контролю доступу до даних конкретної філії, що реалізується через метод перевірки прав у BranchService. Проектування передбачає, що кожна дія з даними клієнта супроводжується створенням запису в історії через ClientHistory, що є критично важливим для подальшого аналізу маркетингової активності та поведінки споживачів. Це забезпечує повну прозорість операцій та дозволяє відновити хронологію взаємодії з контрагентом у разі виникнення суперечливих ситуацій.

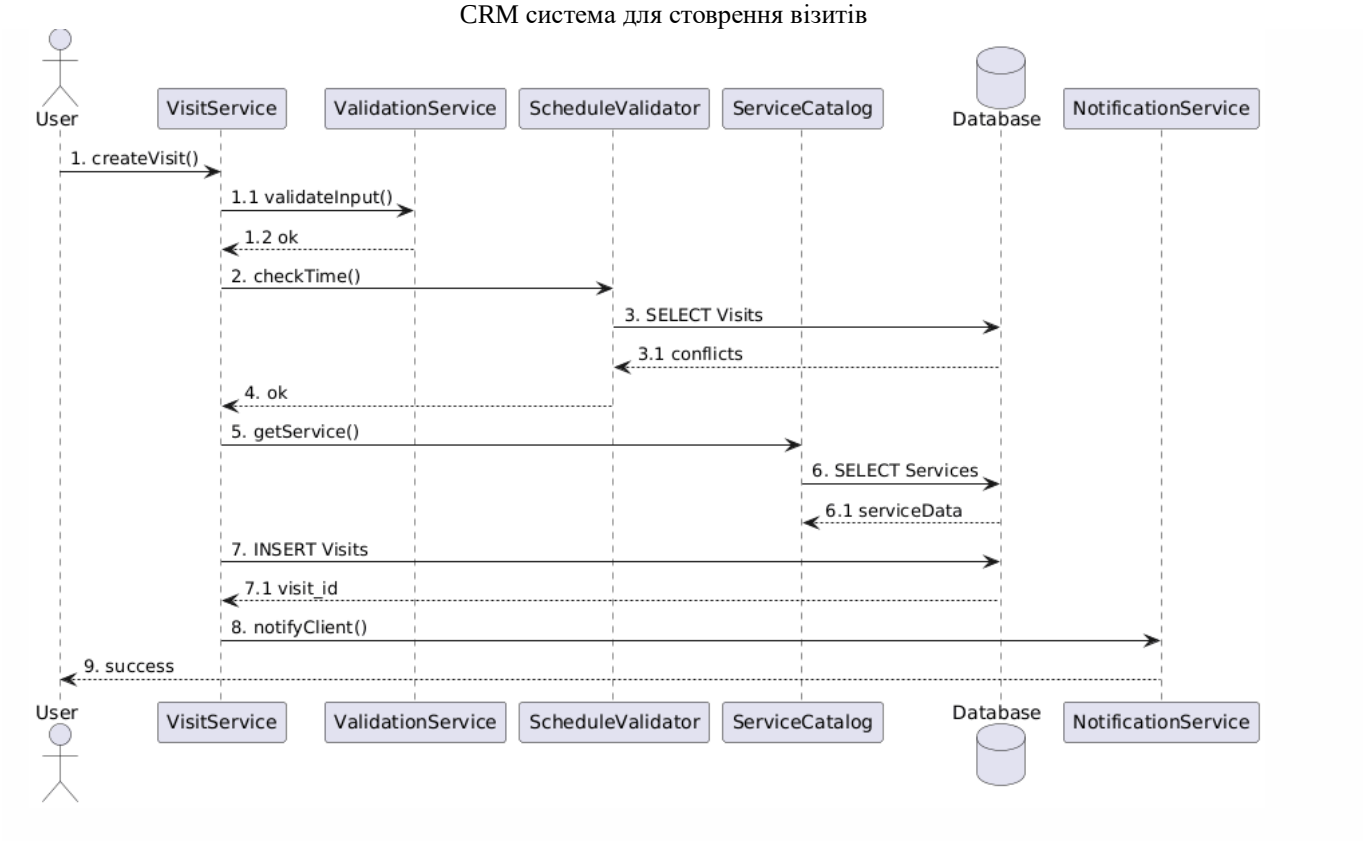


Рисунок 3.5 - Створення візиту

Найбільш навантаженим сценарієм є створення візиту рисунок 3.5, оскільки він вимагає синхронізації графіків багатьох об’єктів. VisitService взаємодіє зі спеціалізованим валідатором розкладу, який виконує складні запити до бази даних для виявлення часових конфліктів. Паралельно система звертається до каталогу послуг для отримання актуальних цін та нормативів тривалості процедур. Тільки після успішного проходження всіх етапів перевірки відбувається транзакційний запис у базу даних та відправка підтвердження клієнту. Такий підхід гарантує неможливість подвійного бронювання одного і того самого часу для різних відвідувачів.

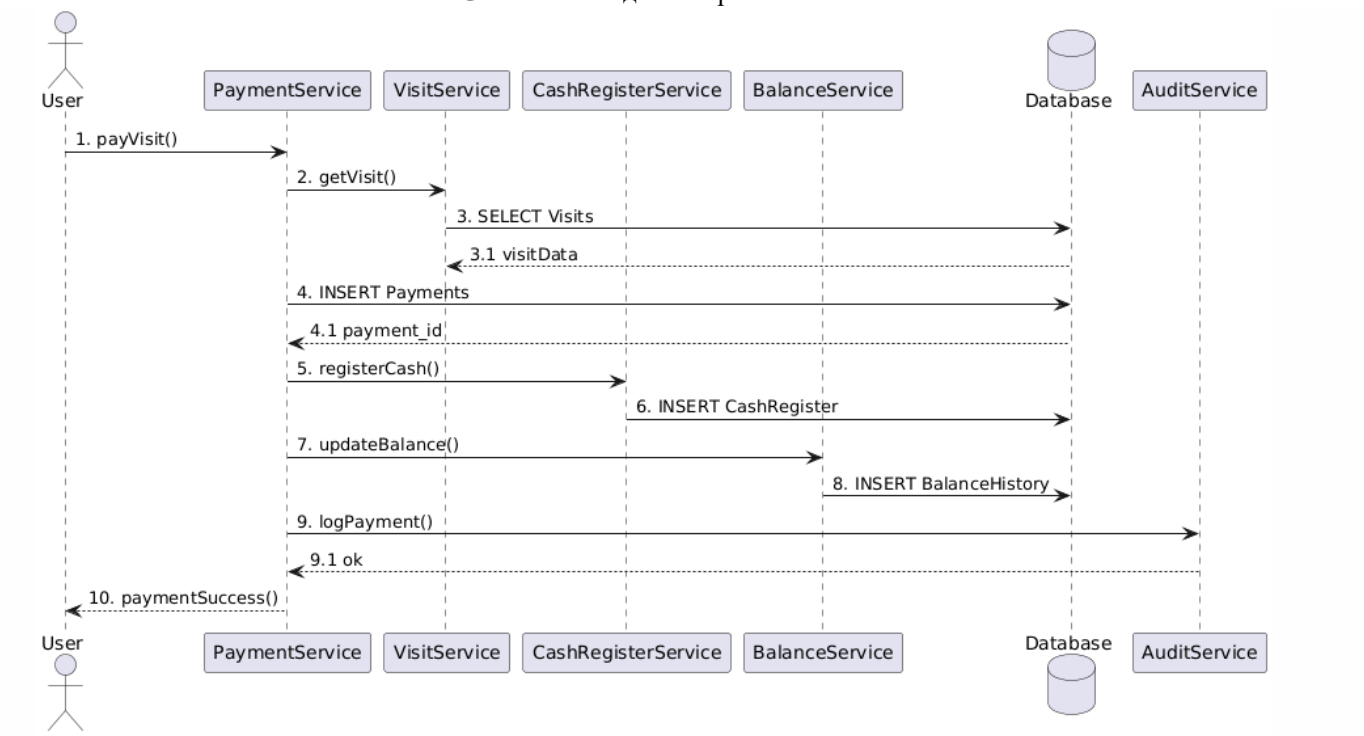


Рисунок 3.6 – Оплата візиту

Фінансовий блок системи, представлений сценарієм оплати візиту рисунок 3.6, базується на принципі суворої звітності. PaymentService виступає головним контролером, який послідовно оновлює статуси в системі обліку візитів, реєструє надходження коштів у касовому модулі та коригує баланси відповідних рахунків. Кожна фінансова операція дублюється в сервісі аудиту, що створює захищений від модифікацій слід транзакцій. Це дозволяє власнику бізнесу бути впевненим у достовірності фінансової звітності та унеможливлює приховування прибутку персоналом.

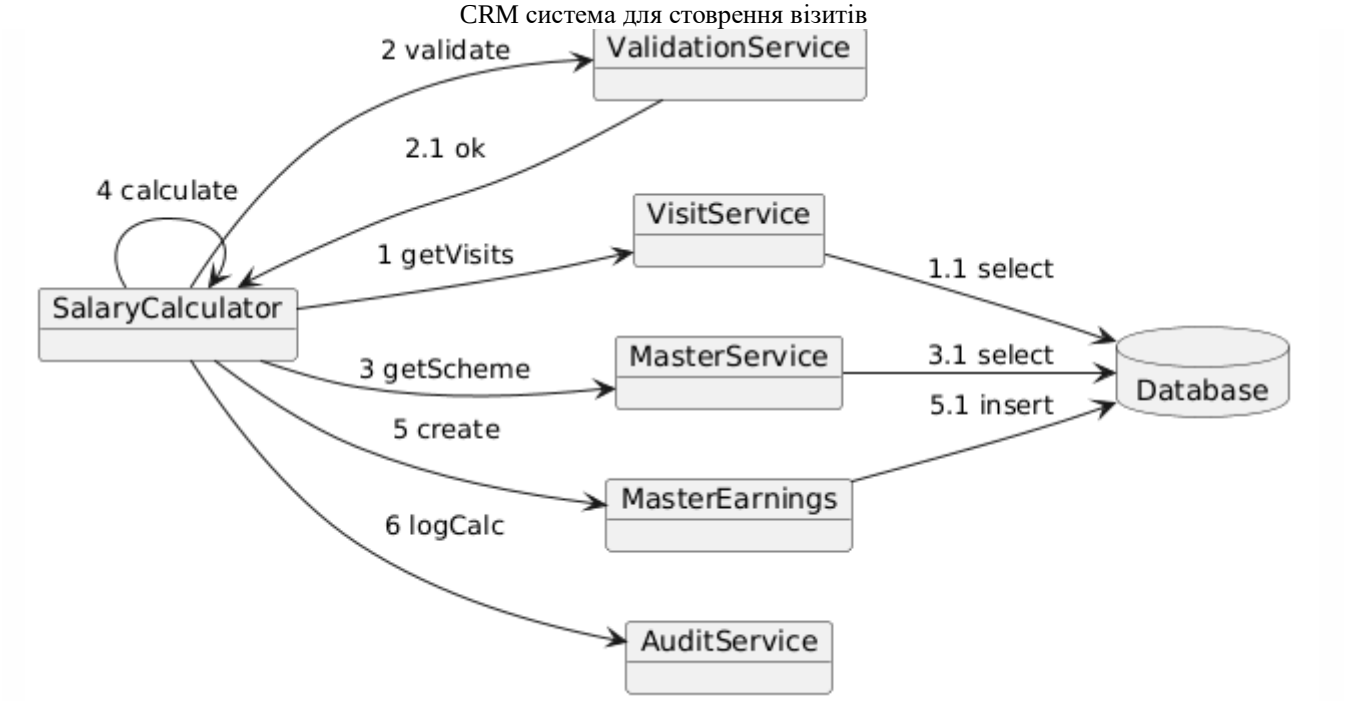


Рисунок 3.7 - Нарахування зарплати майстру

На завершальному етапі розглядається автоматизація розрахунків із персоналом рисунок 3.7. SalaryCalculator агрегує дані про всі виконані та оплачені візити майстра за певний період. Завдяки інтеграції з MasterService система отримує індивідуальні налаштування схем нарахувань, що дозволяє гнучко враховувати відсотки від послуг або фіксовані ставки. Результати розрахунків фіксуються в модулі MasterEarnings, що забезпечує автоматичне формування відомостей на виплату заробітної плати. Описана динамічна модель підтверджує, що архітектура системи є масштабованою, надійною та відповідає сучасним вимогам до проектування корпоративного програмного забезпечення.

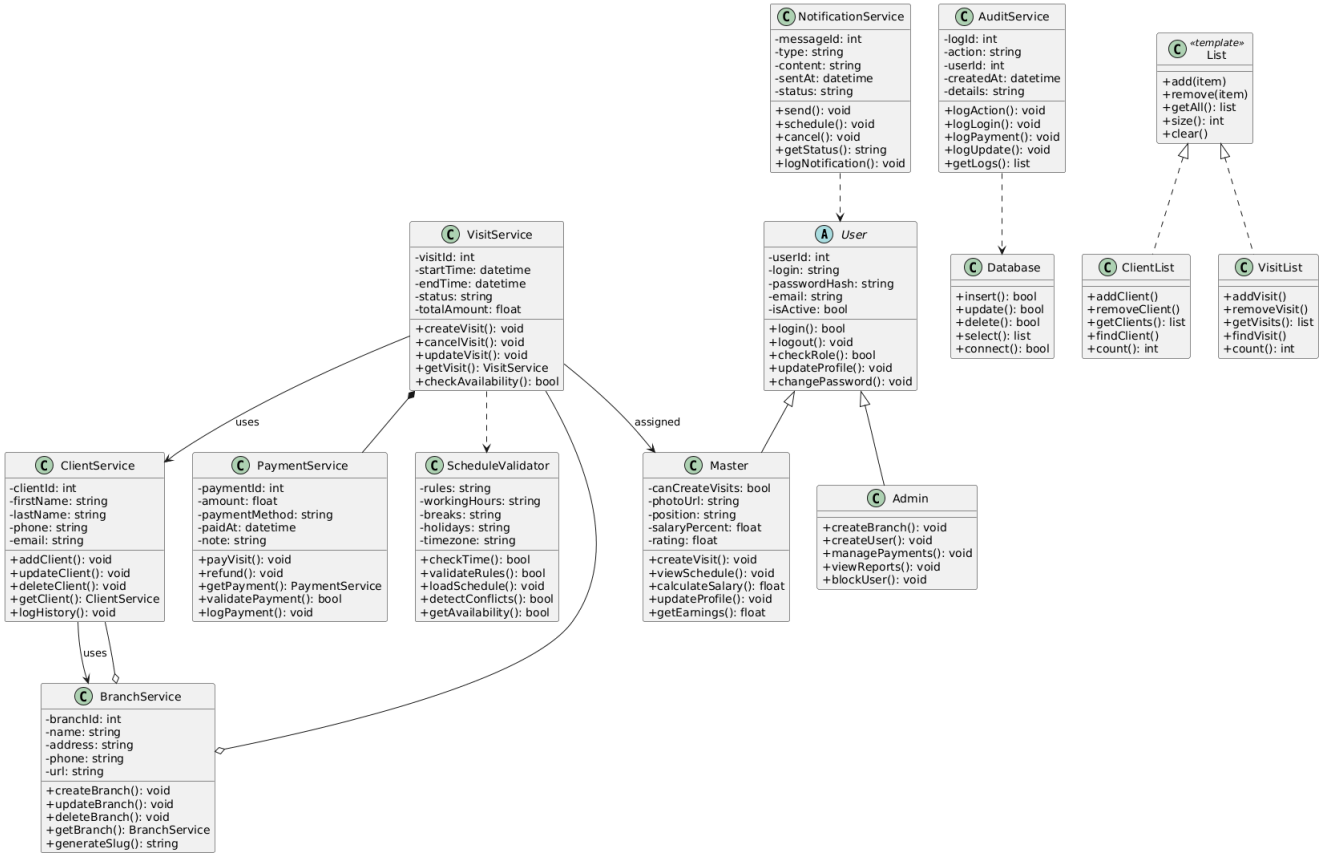


Рисунок 3.8 – Діаграмма класів

Центральним елементом архітектурного проектування CRM-системи є діаграмма класів рисунок 3.8, яка визначає логічну організацію програмного коду та структуру даних. Архітектура побудована за принципами об’єктно-орієнтованого проектування та розділена на функціональні блоки, що забезпечують модульність та високу згуртованість системи. В основі ієрархії користувачів лежить абстрактний клас User, який містить базові атрибути автентифікації, такі як ідентифікатор, логін, хеш пароля та електронну пошту. Від нього успадковуються конкретні класи реалізації: Admin, що володіє розширеними правами на створення філій та керування користувачами, та Master, який доповнений професійними атрибутами, включаючи рейтинг, посаду та відсоток заробітної плати. Така реалізація дозволяє гнучко розмежовувати рівні доступу та функціональні можливості персоналу.

Ключовим операційним вузлом системи є клас VisitService, який агрегує інформацію про візити та координує взаємодію між іншими сервісами. Він володіє

атрибутами часових рамок та статусу візиту, а його методи забезпечують повний цикл управління записами. VisitService має безпосередні зв’язки з ClientService для ідентифікації клієнтів та з PaymentService для обробки фінансових транзакцій. Особливу роль відіграє асоціація з класом ScheduleValidator, який відповідає за складну логіку перевірки робочих годин, святкових днів та виявлення часових конфліктів у розкладі майстрів. Ця структура гарантує цілісність бізнес-процесу бронювання та запобігає помилкам планування на рівні програмної логіки.

Управління географічною структурою бізнесу реалізовано через клас BranchService, який зберігає інформацію про філії, включаючи контактні дані та унікальні URL-адреси. Зв’язок типу композиції між BranchService та іншими сервісами вказує на те, що більшість операцій у системі жорстко прив’язані до контексту конкретної локації. Для підтримки чистоти архітектури використано патерн проектування «Список» (шаблонний клас List), від якого ініціалізуються спеціалізовані колекції ClientList та VisitList. Це дозволяє стандартизувати операції додавання, пошуку та підрахунку елементів у списках клієнтів та візитів, спрощуючи підтримку коду та масштабування системи.

Забезпечення надійності та прозорості операцій покладено на службові класи NotificationService та AuditService. Перший спеціалізується на управлінні чергою повідомлень, відстеженні їхнього статусу та термінів відправки, що є критичним для сервісної індустрії. Другий виконує роль глобального реєстратора подій, фіксуючи кожну зміну даних, вхід користувача або фінансовий платіж у системному журналі. Взаємодія всіх сервісів із фізичним рівнем збереження даних реалізується через клас Database, який інкапсулює логіку SQL-запитів та з’єднання зі сховищем. Описана статична модель демонструє високий рівень деталізації та готовність архітектури до реалізації на об’єктно-орієнтованій мові програмування з чітким дотриманням принципів розділення відповідальності та безпеки даних.

3.2 Проєктування графічного інтерфейсу користувача

На етапі проектування візуальної оболонки CRM-системи основний фокус було зосереджено на створенні інтуїтивно зрозумілого та мінімалістичного

інтерфейсу, що мінімізує когнітивне навантаження на персонал. Дизайн системи базується на принципах компонентного підходу, де кожен елемент інтерфейсу відповідає за конкретну бізнес-функцію, описану в попередніх розділах.

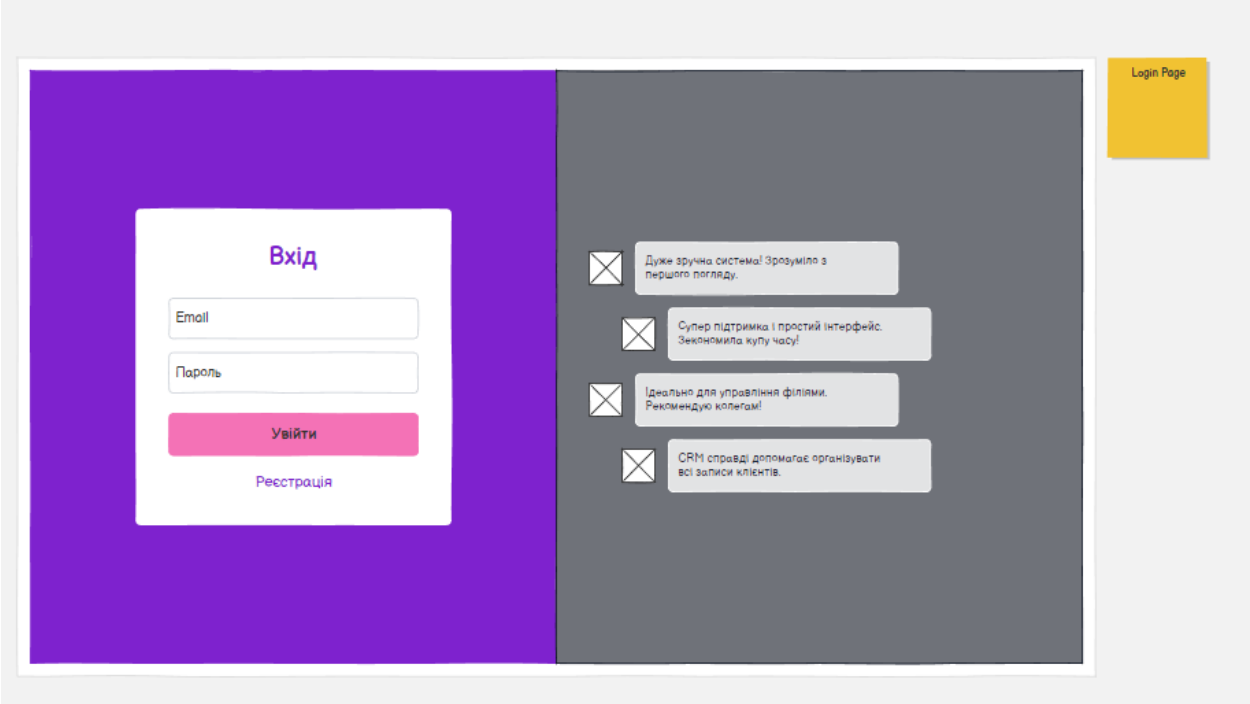


Рисунок 3.9 – Mockup вхід у систему

Макет сторінки авторизації рисунок 3.9 розділяти на дві функціональні зони. У лівій частині на насиченому фіолетовому фоні розміщувати білу форму входу. Використовувати текстові поля для введення електронної пошти та пароля. Передбачати яскраву рожеву кнопку для підтвердження дії та текстове посилання для переходу до реєстрації. У правій частині на темно-сірому фоні розташовувати блок із відгуками користувачів. Кожен відгук оформлювати у вигляді світлої текстової плашки поруч із графічною іконкою профілю. Використовувати сучасний беззасічковий шрифт та дотримуватися контрастної колірної гами для виділення ключових елементів інтерфейсу.

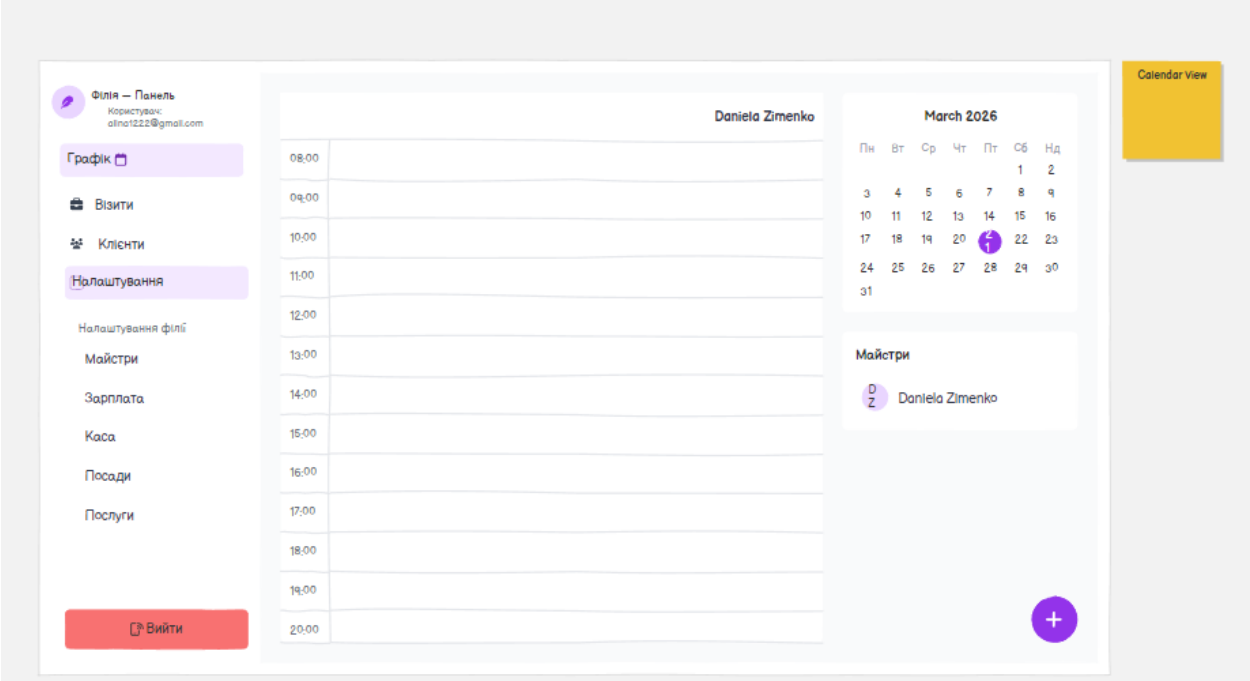


Рисунок 3.10 – Mockup Графік роботи

Макет сторінки календаря рисунок 3.10 проєктувати як багатопанельну робочу область для управління графіком записів. У лівій частині розміщувати вертикальну панель навігації з логотипом, даними користувача та основним меню, що включає розділи графіка, візитів, клієнтів та налаштувань. У нижньому куті навігаційної панелі передбачати контрастну кнопку для виходу з системи. Центральну частину інтерфейсу відводити під детальний розклад дня з погодинною сіткою від 08:00 до 20:00 та відображенням імені відповідального майстра. Праву частину екрана розділяти на два допоміжні блоки: інтерактивний календар на поточний місяць для швидкої навігації по датах та список доступних майстрів. У правому нижньому куті робочої області розташовувати яскраву круглу кнопку з символом плюса для оперативного додавання нових подій. Оформлювати інтерфейс у світлих тонах із використанням фіолетових акцентів для активних елементів та дотримуватися мінімалістичного стилю для полегшення сприйняття інформації.

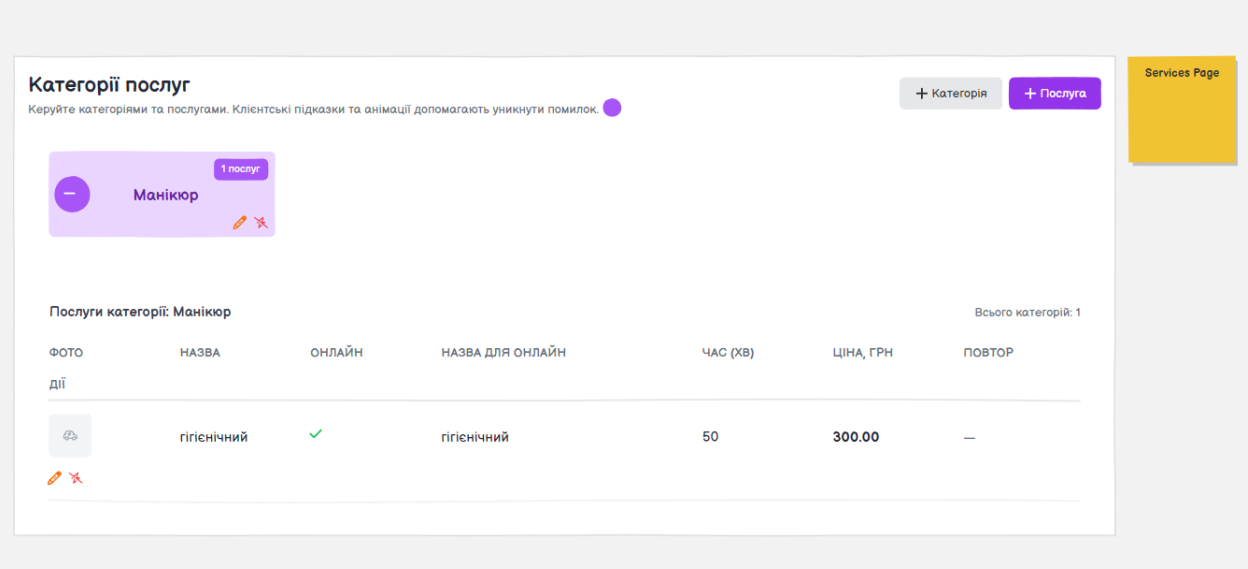


Рисунок 3.11 - Mockup Налаштування графіка послуг

Макет сторінки категорій та послуг рисунок 3.11 представляти як інструмент для адміністративного керування прейскурантом. У верхній частині робочої області розташовувати заголовок із коротким описом функціонала та кнопками для додавання нових категорій і конкретних послуг. Основну частину екрана відводити під інтерактивні картки категорій, які містять назву, кількість прикріплених позицій та елементи для редагування або видалення. Нижче розміщувати таблицю з детальним переліком послуг обраної категорії, що включає стовпці для фото, назви, статусу онлайн-запису, тривалості виконання та вартості у гривнях. Передбачати можливість швидкого редагування кожної позиції за допомогою допоміжних іконок дій. Дотримуватися цілісного візуального стилю з використанням фіолетових акцентів для кнопок та активних станів елементів на світлому мінімалістичному фоні.

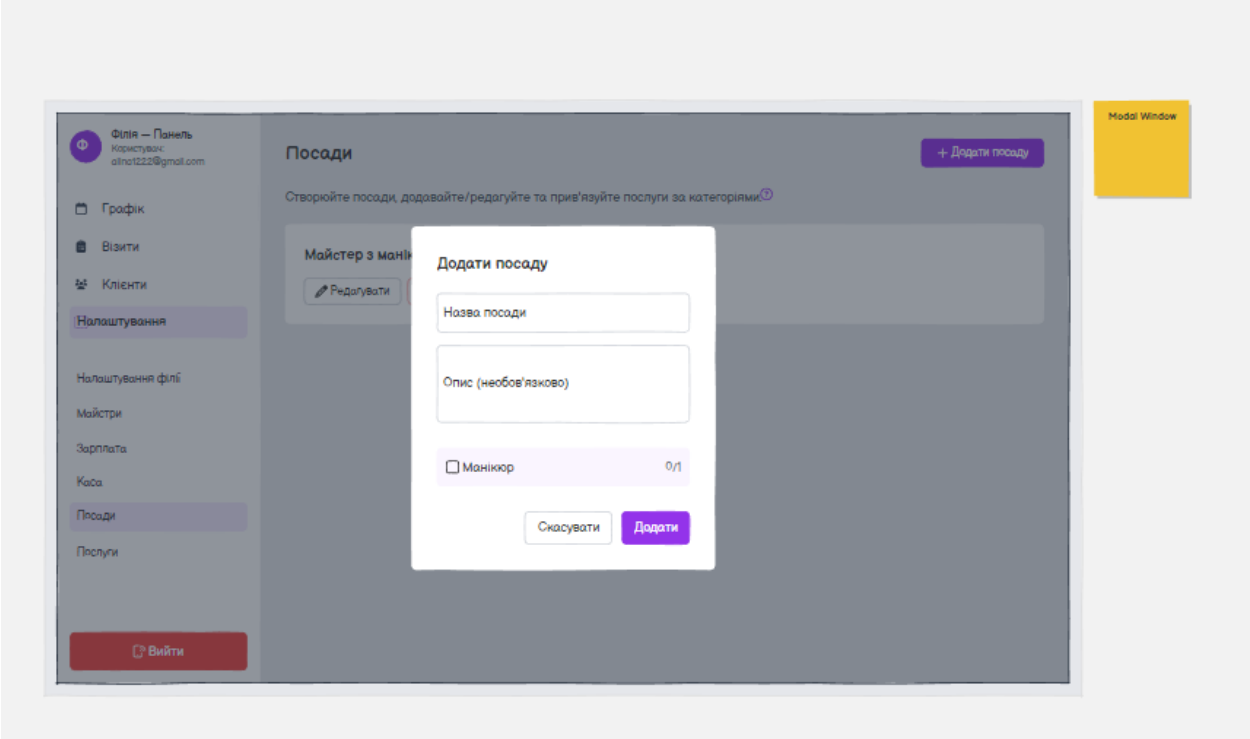


Рисунок 3.12 – Mockup Налаштування посад

Макет модального вікна додавання посади рисунок 3.12 проектувати як допоміжний шар інтерфейсу, що з’являється поверх основної сторінки «Посади». Візуально виділяти вікно за допомогою білої картки з м'якими тінями на фоні затемненого основного контенту. У верхній частині вікна розміщувати заголовок із чітким описом дії. Для введення даних використовувати текстові поля для назви посади та необов’язкового опису. Передбачати блок вибору категорій із чекбоксами для прив'язки конкретних послуг до нової посади. У нижній правій частині вікна розташовувати дві функціональні кнопки: «Скасувати» для закриття вікна без збереження та «Додати» у фіолетовому кольорі для підтвердження створення запису. Дотримуватися загальної стилістики системи через використання заокруглених кутів, ідентичних шрифтів та фірмової колірної палітри.

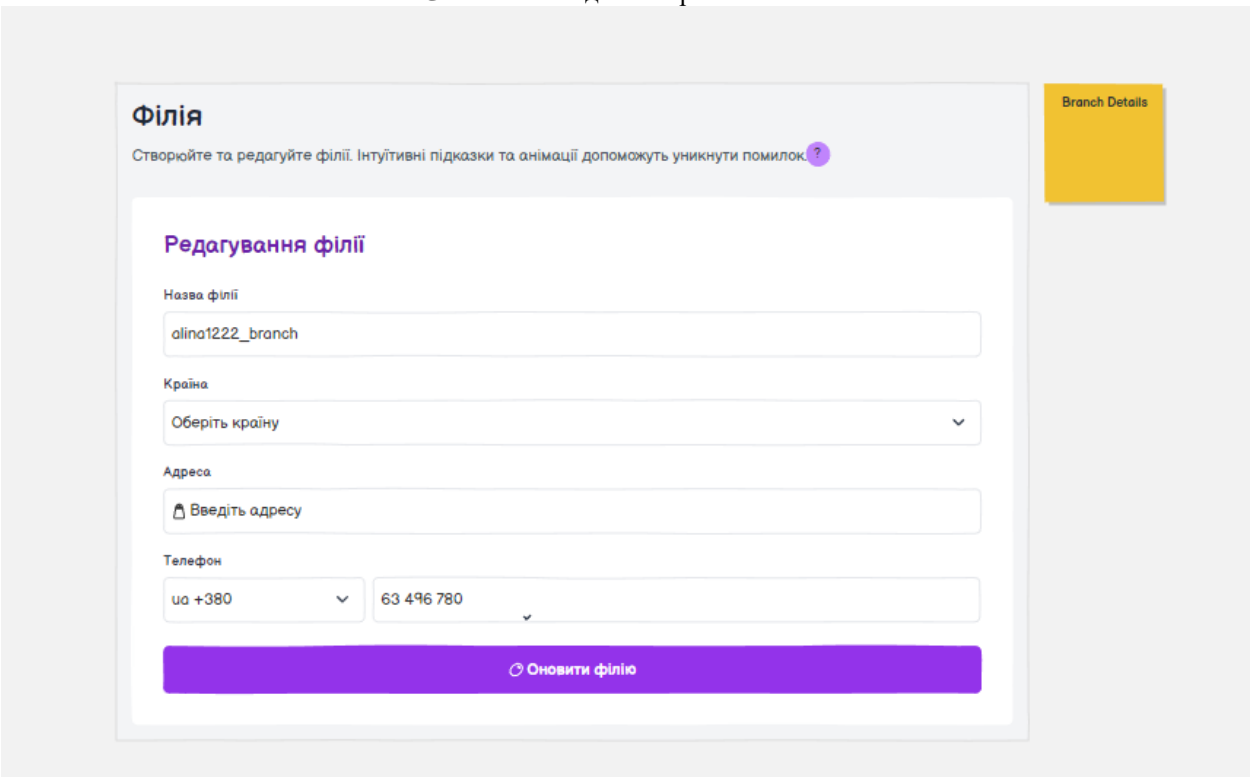


Рисунок 3.13 - Mockup Налаштування філії

Макет сторінки редагування даних філії рисунок 3.13 представляти як форму для налаштування основної інформації про підрозділ. У верхній частині сторінки розташовувати заголовок «Філія» з коротким поясненням щодо можливості створення та редагування об'єктів. Основну частину інтерфейсу відводити під білу картку з назвою секції «Редагування філії». Для збору даних використовувати послідовність текстових полів: назву філії, випадний список для вибору країни, поле для введення адреси з іконкою локації та розділене поле для контактного номера телефону з кодом країни. У нижній частині форми передбачати акцентну фіолетову кнопку з написом «Оновити філію» для збереження внесених змін. Дотримуватися загального стилю системи, що базується на значних відступах, чіткій типографіці та лаконічному дизайні елементів введення.

Висновки до розділу 3

У третьому розділі було виконано проєктування та реалізацію CRM-системи для автоматизації діяльності салонів краси та сервісних центрів. На основі аналізу бізнес-процесів визначено основні ролі користувачів, їх функціональні

можливості та сценарії взаємодії із системою. Для формалізації вимог побудовано UML-діаграму прецедентів, яка відображає розподіл прав доступу між адміністратором, головною філією та майстром.

З метою деталізації динамічних процесів функціонування системи розроблено діаграми послідовності та комунікацій для ключових сценаріїв роботи: авторизації користувача, створення філії, реєстрації клієнта, формування візиту, проведення оплати та нарахування заробітної плати майстрам. Побудовані моделі дозволили визначити порядок взаємодії між програмними компонентами та забезпечити цілісність бізнес-логіки системи.

У процесі архітектурного проєктування сформовано діаграму класів, яка відображає структуру програмного забезпечення, взаємозв’язки між об’єктами та принципи розподілу відповідальності між модулями. Запропонована архітектура базується на принципах об’єктно-орієнтованого програмування, забезпечує модульність, масштабованість та можливість подальшого розширення функціоналу системи.

Окрему увагу приділено проєктуванню графічного інтерфейсу користувача. Розроблені макети основних сторінок і модальних вікон дозволили сформувати єдиний стиль системи, орієнтований на зручність використання, швидкий доступ до функцій та ефективну роботу персоналу. Інтерфейс побудовано відповідно до сучасних принципів UX/UI-дизайну із застосуванням мінімалістичного підходу та інтуїтивно зрозумілої навігації.

Отже, результати проєктування підтверджують, що розроблена CRM-система повністю відповідає поставленим функціональним вимогам, забезпечує автоматизацію основних бізнес-процесів підприємства та створює надійну основу для подальшої реалізації, тестування й впровадження програмного продукту в реальних умовах експлуатації.

4 РОЗРОБКА ТА ПРОГРАМНА РЕАЛІЗАЦІЯ CRM-СИСТЕМИ УПРАВЛІННЯ ВІЗИТАМИ

4.1 Специфікація програмного забезпечення

У межах розробки інформаційної системи управління візитами CRM було спроєктовано та реалізовано реляційну базу даних на основі СУБД Microsoft SQL[17] Server. Вибір даної системи управління базами даних обумовлений її високою продуктивністю, підтримкою складних зв’язків між таблицями, можливістю забезпечення цілісності даних за рахунок використання зовнішніх ключів, а також розвиненим механізмом індексації та транзакційної обробки.

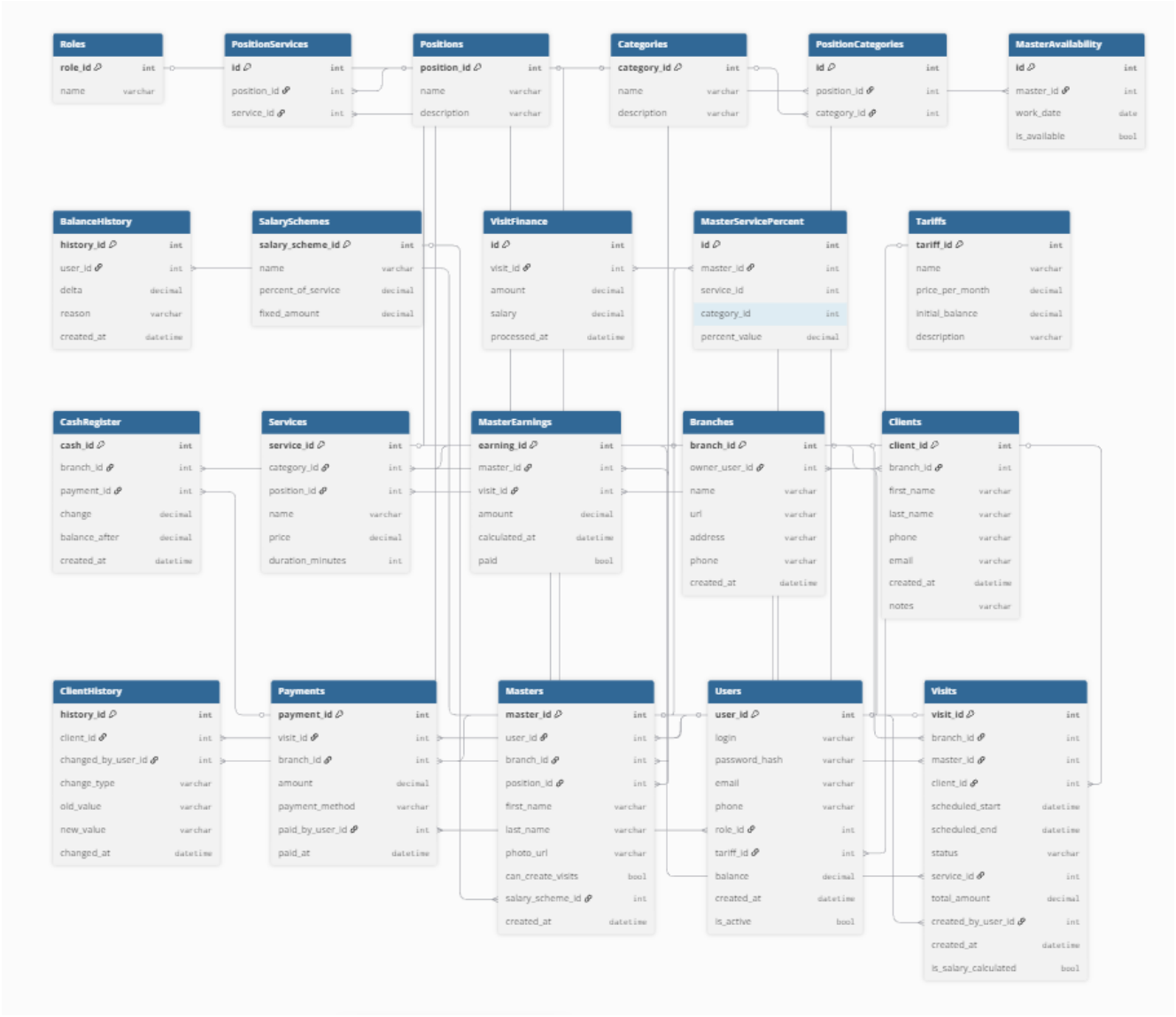


Рисунок 4.1 – Діаграма бази даних

Структура бази даних побудована відповідно до принципів нормалізації та орієнтована на мінімізацію надлишковості даних. Основою інформаційної моделі є сутність користувачів системи, яка реалізована у вигляді таблиці Users рисунок 4.2. Дана таблиця містить ідентифікатор користувача, який виступає первинним ключем, а також поля для зберігання логіну, пароля, електронної пошти, телефону, балансу та статусу активності. Для забезпечення розмежування прав доступу використовується зовнішній ключ, який пов’язує користувача з таблицею Roles, що визначає роль у системі, а також з таблицею Tariffs, яка містить інформацію про тарифний план та фінансові умови використання системи.

Для організації структури бізнесу в системі передбачено сутність філій, реалізовану через таблицю Branches. Кожна філія пов’язана з користувачем-власником за допомогою зовнішнього ключа, що забезпечує логічну прив’язку бізнес-структури до конкретного користувача системи. Додатково зберігаються атрибути, що характеризують назву, адресу, контактні дані та службову URL-адресу філії.

Працівники системи, тобто майстри, представлені таблицею Masters, яка містить інформацію про користувача, філію та посаду. Зв’язки між майстром і відповідними сутностями реалізовані через зовнішні ключі, що посилаються на таблиці Users, Branches та Positions. Окрім цього, передбачено зв’язок із таблицею SalarySchemes, яка визначає схему нарахування заробітної плати у вигляді фіксованої суми або відсотка від вартості послуг.

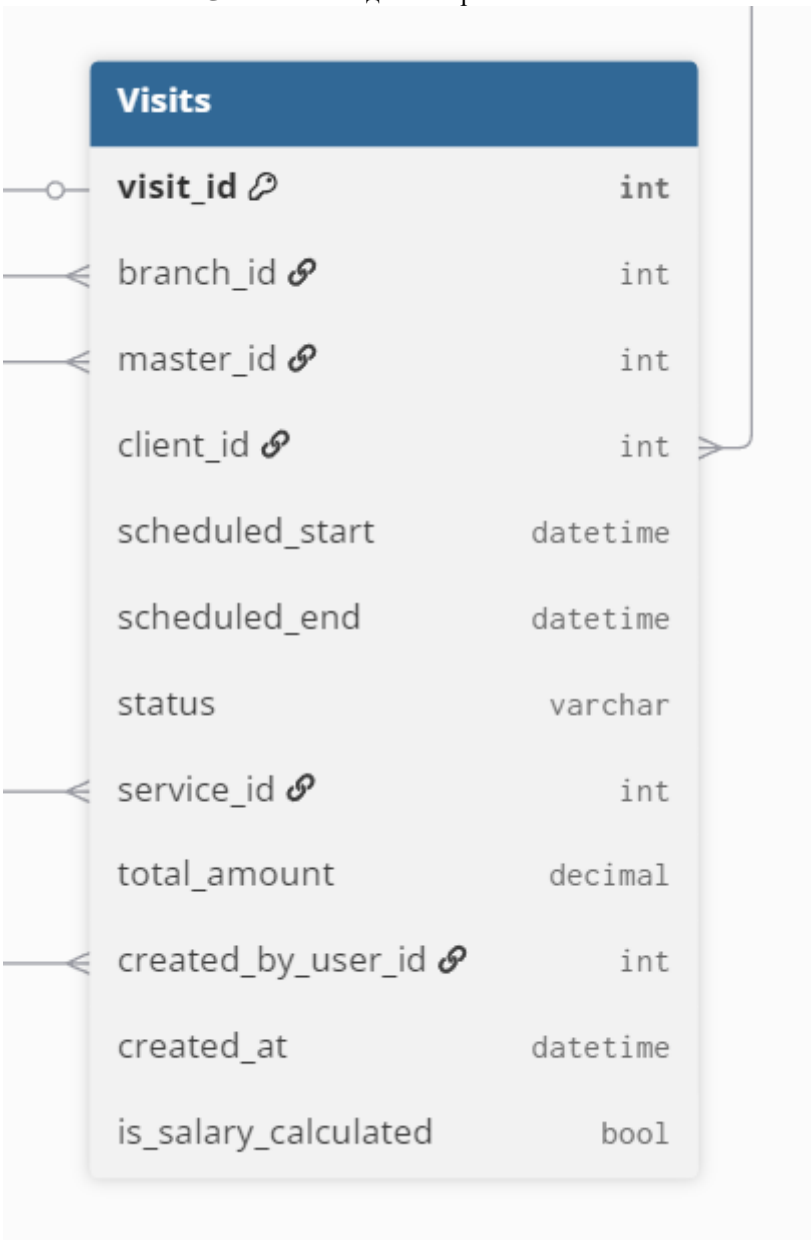


Рисунок 4.2 – Основна таблиця Users

Клієнти системи зберігаються у таблиці Clients, яка має зв’язок із філією, що дозволяє розділяти клієнтські дані між різними бізнес-структурами. Для підвищення ефективності пошуку та обробки даних у цій таблиці додатково створено індекси за телефонним номером та прізвищем клієнта, що забезпечує пришвидшення вибірки при фільтрації даних.

Послуги та їх класифікація реалізовані через таблиці Services та Categories, між якими встановлено зв’язок через зовнішній ключ. Така структура дозволяє групувати послуги за категоріями та забезпечує гнучке розширення системи без

зміни її основної архітектури. Додатково реалізовано зв’язок між послугами та посадами, що дозволяє обмежувати доступність певних послуг залежно від ролі або кваліфікації працівника.

Основною операційною сутністю системи є таблиця Visits, яка зберігає інформацію про запис клієнта на послугу. У цій таблиці фіксуються дані про філію, майстра, клієнта, послугу, запланований час початку та завершення, статус візиту, а також загальна сума. Зв’язки з іншими сутностями реалізовані через зовнішні ключі, що забезпечує цілісність даних та виключає можливість створення некоректних записів.

Фінансова частина системи реалізована через таблиці Payments, CashRegister та MasterEarnings. Таблиця Payments відповідає за збереження інформації про здійснені платежі, включаючи суму, метод оплати та прив’язку до конкретного візиту. На основі цих даних формується касовий облік у таблиці CashRegister, яка фіксує зміни балансу філії. Додатково таблиця MasterEarnings використовується для обліку нарахувань майстрам за виконані послуги.

Для забезпечення аудиту та контролю змін у системі реалізовано додаткові таблиці історії, зокрема BalanceHistory та ClientHistory, які зберігають інформацію про зміни балансу користувачів та редагування даних клієнтів. Для автоматизації фіксації змін у даних клієнтів використовується тригер, який при оновленні записів автоматично зберігає попередні та нові значення у відповідній історичній таблиці.

Таким чином, розроблена структура бази даних забезпечує повну підтримку функціоналу інформаційної системи, включаючи управління користувачами, бізнес-структурою, клієнтами, послугами, візитами та фінансовими операціями, а також гарантує цілісність і узгодженість даних завдяки використанню первинних та зовнішніх ключів, індексів і механізмів автоматичного контролю змін.

4.2 Реалізація серверної частини та модулів

У межах розробки інформаційної системи CRM для управління візитами було реалізовано серверну частину застосунку з використанням мови програмування PHP та СУБД Microsoft SQL Server[15]. Взаємодія з базою даних здійснюється

через драйвер sqlsrv, що забезпечує безпечне виконання параметризованих SQL-запитів та запобігає SQL-ін’єкціям.

Основним модулем серверної частини є підсистема автентифікації та реєстрації користувачів, яка забезпечує створення нових облікових записів, перевірку унікальності даних, а також вхід користувача в систему з подальшим збереженням сесійної інформації.

При реалізації функціоналу реєстрації рисунок 4.3 користувача здійснюється отримання вхідних даних з HTTP-запиту, таких як логін, електронна пошта, номер телефону, пароль та обраний тарифний план. Перед створенням нового запису виконується перевірка на наявність користувача з аналогічним логіном або email у таблиці Users, що дозволяє забезпечити унікальність облікових записів.

```php
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
  $login = trim($_POST['login']);
  $email = trim($_POST['email']);
  $phone = trim($_POST['phone']);
  $password = $_POST['password'];
  $selected_tariff = $_POST['tariff_id'];

  $tariff_map = [1 => 3, 2 => 4];
  $tariff_id = $tariff_map[$selected_tariff] ?? 3;

  // Проверяем, существует ли пользователь
  $sqlCheck = "SELECT COUNT(*) AS count FROM Users WHERE login = ? OR email = ?";
  $stmtCheck = sqlsrv_query($conn, $sqlCheck, [$login, $email]);
  if ($stmtCheck === false) {
      die("SQL Check Error: " . print_r(sqlsrv_errors(), true));
  }

  $row = sqlsrv_fetch_array($stmtCheck, SQLSRV_FETCH_ASSOC);
  if ($row['count'] > 0) {
      $message = "Логин или email уже используется.";
  } else {

      $sqlTariff = "SELECT initial_balance FROM Tariffs WHERE tariff_id = ?";
      $stmtTariff = sqlsrv_query($conn, $sqlTariff, [$tariff_id]);
      if ($stmtTariff === false) {
          die("SQL Tariff Error: " . print_r(sqlsrv_errors(), true));
      }

      $tariffRow = sqlsrv_fetch_array($stmtTariff, SQLSRV_FETCH_ASSOC);
      $balance = $tariffRow ? $tariffRow['initial_balance'] : 0;


      $hashedPassword = password_hash($password, PASSWORD_DEFAULT);


      $sqlInsert = "INSERT INTO Users (login, password_hash, email, phone, role_id, tariff_id, balance)
                    OUTPUT INSERTED.user_id
                    VALUES (?, ?, ?, ?, ?, ?, ?)";
      $stmtInsert = sqlsrv_query($conn, $sqlInsert, [$login, $hashedPassword, $email, $phone, 1, $tariff_id, $balance]);

      if ($stmtInsert === false) {
          die("SQL Insert User Error: " . print_r(sqlsrv_errors(), true));
      }
```

Рисунок 4.3 – Реєстрація користувача

Після цього відбувається отримання початкового балансу, який залежить від обраного тарифного плану, шляхом звернення до таблиці Tariffs. Пароль користувача перед збереженням у базі даних хешується з використанням функції password_hash, що підвищує рівень безпеки зберігання конфіденційних даних.

Далі виконується вставка нового користувача в таблицю Users із зазначенням ролі за замовчуванням та прив’язаного тарифу. Одразу після створення облікового запису автоматично формується запис у таблиці Branches, який відповідає персональній філії користувача, що дозволяє пов’язати бізнес-структуру з конкретним власником.

Окремо реалізовано модуль авторизації користувача рисунок 4.4 . Під час входу в систему виконується пошук користувача за електронною поштою, після чого здійснюється перевірка пароля за допомогою функції password_verify. У разі успішної автентифікації в сесію користувача зберігаються основні параметри, такі як email, логін та роль користувача, що дозволяє реалізувати механізм розмежування доступу в межах системи.

```php
$db = new NewsDB();
$db->connect();
$conn = $db->getConnection();

$message = '';

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
  $email = trim($_POST['email']);
  $password = $_POST['password'];


  $sql = "SELECT * FROM Users WHERE email = ?";
  $params = [$email];
  $stmt = sqlsrv_prepare($conn, $sql, $params);

  if(!$stmt) die(print_r(sqlsrv_errors(), true));

  sqlsrv_execute($stmt);

  $user = sqlsrv_fetch_array($stmt, SQLSRV_FETCH_ASSOC);

  if ($user) {
    
    if (password_verify($password, $user['password_hash'])) {

      $_SESSION['email'] = $user['email'];
      $_SESSION['login'] = $user['login'];
      $_SESSION['role_id'] = $user['role_id'];

      $sqlBranch = "SELECT * FROM Branches WHERE owner_user_id = ?";
      $stmtBranch = sqlsrv_query($conn, $sqlBranch, [$user['id']]);
      $branch = sqlsrv_fetch_array($stmtBranch, SQLSRV_FETCH_ASSOC);

      if ($branch) {
          $_SESSION['branch_id'] = $branch['id'];
      }

      header("Location: branch.php");
      exit();
    } else {
      $message = "Неправильний email або пароль!";
    }
  } else {
    $message = "Користувача не знайдено!";
  }
}
```

Рисунок 4.4 – Логінізація

Також у процесі авторизації здійснюється отримання інформації про філію користувача з таблиці Branches, що дозволяє прив'язати подальші дії користувача до конкретної бізнес-структури.

Інтерфейс користувача реалізований у вигляді веб-сторінок з використанням HTML, CSS та JavaScript. На головній сторінці реалізовано модальне вікно реєстрації, яке дозволяє вводити дані користувача без переходу на окрему сторінку, що покращує зручність використання системи.

Одним із основних є календар обробки візитів та автоматичного фінансового облік. На початку роботи скрипта виконується ініціалізація сесії користувача через session_start(); рисунок 4.6, після чого перевіряється факт авторизації. Якщо користувач не авторизований, виконується перенаправлення на сторінку входу, що забезпечує базовий рівень захисту системи.

```php
<?php
session_start();
require_once 'db.php';

if (!isset($_SESSION['email'])) {
  header("Location: login.php");
  exit();
}
require_once 'toast.php';
$db = new NewsDB();
$db->connect();
$conn = $db->getConnection();
function stop($msg) {
  echo "<script>showToast(".json_encode($msg).", 'error');</script>";
  exit();
}
// === USER & BRANCH ===
$user_email = $_SESSION['email'];
$sqlUser = "SELECT user_id FROM Users WHERE email = ?";
$stmtUser = sqlsrv_query($conn, $sqlUser, [$user_email]);
$user = sqlsrv_fetch_array($stmtUser, SQLSRV_FETCH_ASSOC);
if (!$user) die("Користувача не знайдено");

$sqlBranch = "SELECT TOP 1 branch_id FROM Branches WHERE owner_user_id = ?";
$stmtBranch = sqlsrv_query($conn, $sqlBranch, [$user['user_id']]);
$branch = sqlsrv_fetch_array($stmtBranch, SQLSRV_FETCH_ASSOC);
if (!$branch) die("У користувача немає філії");
$branch_id = $branch['branch_id'];

// === LOAD DATA ===
$masters = [];
$stmt = sqlsrv_query($conn, "
  SELECT m.master_id, m.first_name, m.last_name, m.position_id, p.name AS position_name
  FROM Masters m
  JOIN Positions p ON m.position_id = p.position_id
  WHERE m.branch_id = ?
", [$branch_id]);


while ($r = sqlsrv_fetch_array($stmt, SQLSRV_FETCH_ASSOC)) $masters[] = $r;

```

Рисунок 4.5 – Первірка на користувача та досутп до календаря

Після цього встановлюється підключення до бази даних через створений клас NewsDB, який інкапсулює логіку з’єднання. Отримання об’єкта з’єднання виконується через виклик $conn = $db->getConnection();, що дозволяє уніфіковано працювати з SQL Server у всьому модулі.

Далі система визначає поточного користувача за email із сесії. Це реалізовано запитом до таблиці користувачів, наприклад: SELECT user_id FROM Users WHERE email = ? рисунок 4.5. На основі отриманого ідентифікатора визначається філія користувача через запит до таблиці філій, що дозволяє ізолювати дані між різними відділеннями системи.

Після ідентифікації філії завантажуються основні дані, необхідні для роботи інтерфейсу. Зокрема, список майстрів отримується через SQL-запит до таблиці Masters із приєднанням таблиці посад, що дозволяє відображати не лише ім’я майстра, а й його спеціалізацію. Аналогічно формується список послуг, які доступні відповідним майстрам, через зв’язок таблиць Services, PositionServices та Masters.

Коли користувач створює новий візит, система обробляє POST-запит з форми. Дані, такі як майстер, послуга, клієнт, час початку та завершення, а також сума, зчитуються зі змінних $_POST. Перед записом у базу виконується базова перевірка, наприклад контроль суми if ($total <= 0) та перевірка наявності клієнта. Після цього формується SQL-запит на вставку нового запису у таблицю візитів, де новий візит створюється зі статусом scheduled.

Окремо реалізовано логіку зміни статусу візиту. При оновленні запису виконується SQL-операція оновлення таблиці Visits, наприклад UPDATE Visits SET status = ?, scheduled_start = ?, scheduled_end = ? WHERE visit_id = ?, що дозволяє змінювати параметри вже створеного запису без його видалення рисунок 4.6.

```php
if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['add_visit'])) {
  $check = sqlsrv_query($conn, "
  SELECT id FROM VisitFinance WHERE visit_id = ?
", [$visit_id]);

if (sqlsrv_fetch_array($check, SQLSRV_FETCH_ASSOC)) {
  die("Этот визит уже оплачен. Редактирование запрещено.");
}
  $master_id = (int)$_POST['master_id'];
  $service_id = (int)$_POST['service_id'];
  $client_id = isset($_POST['client_id']) ? (int)$_POST['client_id'] : null;

  $start = str_replace('T', ' ', $_POST['start_time']);
  $end   = str_replace('T', ' ', $_POST['end_time']);
  $total = (float)$_POST['total_amount'];
  if ($total <= 0) {
    error("Сума візиту повинна бути більше 0");
}
  if (!$client_id) {
      die("Клиент не выбран. Да, без него визит теперь не создаётся.");
  }

  $insert = sqlsrv_query($conn, "
    INSERT INTO Visits
    (branch_id, master_id, client_id, scheduled_start, scheduled_end, status, service_id, total_amount, created_by_user_id)
    VALUES (?, ?, ?, ?, ?, 'scheduled', ?, ?, ?)
  ", [
    $branch_id,
    $master_id,
    $client_id,
    $start,
    $end,
    $service_id,
    $total,
    $user['user_id']
  ]);

  if ($insert) {
    header("Location: calendar.php?msg=OK");
    exit();
  } else {
    die(print_r(sqlsrv_errors(), true));
  }
}
```

Рисунок 4.6 – Створення візиту

Найважливішою частиною модуля є фінансова обробка, яка спрацьовує тільки у випадку, коли статус візиту змінюється на completed. У цей момент система перевіряє, чи був уже оброблений цей візит у таблиці фінансів через запит SELECT id FROM VisitFinance WHERE visit_id = ?. Якщо запис існує, повторна обробка блокується, що запобігає дублюванню нарахувань.

Якщо візит ще не оброблений, система отримує дані про нього та виконує розрахунок доходу майстра. Відсоток винагороди береться з таблиці MasterServicePercent, після чого виконується обчислення заробітку як частини від загальної суми, наприклад: $salary = $amount * ($percent / 100);.

Після цього оновлюється каса філії, де фіксується надходження коштів через вставку запису в таблицю CashRegister, а також створюється запис про дохід

майстра в таблиці MasterEarnings. Завершальним етапом є внесення запису в VisitFinance, який блокує повторну фінансову обробку цього візиту.

Із основних частина коду реалізує логіку створення, редагування та видалення категорій і послуг. При додаванні категорії виконується SQL INSERT у таблицю Categories із передачею назви, опису та branch_id. При додаванні послуги виконується INSERT у таблицю Services, де зберігаються назва послуги, її ціна, тривалість у хвилинах, категорія та branch_id.

Редагування категорій реалізовано через SQL UPDATE, де змінюються поля назви та опису за умовою відповідного category_id і branch_id. Редагування послуг також виконується через UPDATE, при якому оновлюються назва, ціна та тривалість послуги.

Видалення категорії реалізовано через транзакцію рисунок 4.7. Спочатку видаляються всі зв’язки послуг цієї категорії в таблиці PositionServices, після цього видаляються всі послуги з таблиці Services, і в кінці видаляється сама категорія з таблиці Categories. Використання транзакції sqlsrv_begin_transaction та sqlsrv_commit забезпечує цілісність даних і запобігає частковому видаленню інформації.

```php
if (isset($_POST['edit_service'])) {
    $id = intval($_POST['service_id'] ?? 0);
    $name = trim($_POST['name'] ?? '');
    $price = floatval($_POST['price'] ?? 0);
    $duration = intval($_POST['duration'] ?? 0);
    if ($id && $name) {
        sqlsrv_query(
            $conn,
            "UPDATE Services SET name=?, price=?, duration_minutes=? WHERE service_id=? AND branch_id=?",
            [$name, $price, $duration, $id, $branch_id]
        );
    }
}

if (isset($_POST['delete_service'])) {
  $id = intval($_POST['service_id'] ?? 0);
  if ($id) {

      sqlsrv_begin_transaction($conn);

      execOrFail($conn, "DELETE FROM PositionServices WHERE service_id = ?", [$id]);


      execOrFail($conn, "DELETE FROM Services WHERE service_id = ? AND branch_id = ?", [$id, $branch_id]);

      sqlsrv_commit($conn);
  }
}


header("Location: " . $_SERVER['PHP_SELF']);
exit();
}
```

Рисунок 4.7 – Редагування та видалення послуг

Видалення послуги виконується аналогічно: спочатку видаляються зв’язки з таблиці PositionServices, після чого видаляється відповідний запис із таблиці Services.

```php
$categories = [];
$sql = "
    SELECT
        c.category_id,
        c.name AS category_name,
        c.description,
        s.service_id,
        s.name AS service_name,
        s.price,
        s.duration_minutes
    FROM Categories c
    LEFT JOIN Services s ON c.category_id = s.category_id
    WHERE c.branch_id = ?
    ORDER BY c.name, s.name
";
$stmt = sqlsrv_query($conn, $sql, [$branch_id]);

while ($row = sqlsrv_fetch_array($stmt, SQLSRV_FETCH_ASSOC)) {
    $catId = $row['category_id'];
    if (!isset($categories[$catId])) {
        $categories[$catId] = [
            'category_id' => $catId,
            'name' => $row['category_name'],
            'description' => $row['description'],
            'services' => []
        ];
    }
    if ($row['service_id']) {
        $categories[$catId]['services'][] = [
            'service_id' => $row['service_id'],
            'name' => $row['service_name'],
            'price' => $row['price'],
            'duration_minutes' => $row['duration_minutes'],
            'online_booking' => true,
            'online_name' => $row['service_name'],
            'description' => '',
            'repeat_after_days' => 0
        ];
    }
}
$categories = array_values($categories);
?>
```

Рисунок 4.8 – Завантаженя категорій та послуг

Після виконання операцій CRUD формується SQL-запит з об’єднанням таблиць Categories та Services через LEFT JOIN. Отримані дані обробляються у PHP та структуруються у багатовимірний масив, де кожна категорія містить список своїх послуг. Далі цей масив передається на клієнтську сторону у форматі JSON через json_encode.

Клієнтська частина відповідає рисунок 5.8 за динамічне відображення даних. JavaScript будує інтерфейс категорій і послуг без перезавантаження сторінки, дозволяє перемикати категорії, відкривати модальні вікна для додавання та редагування, а також виконувати видалення записів. Додатково реалізовано пошук і фільтрацію категорій на стороні клієнта.

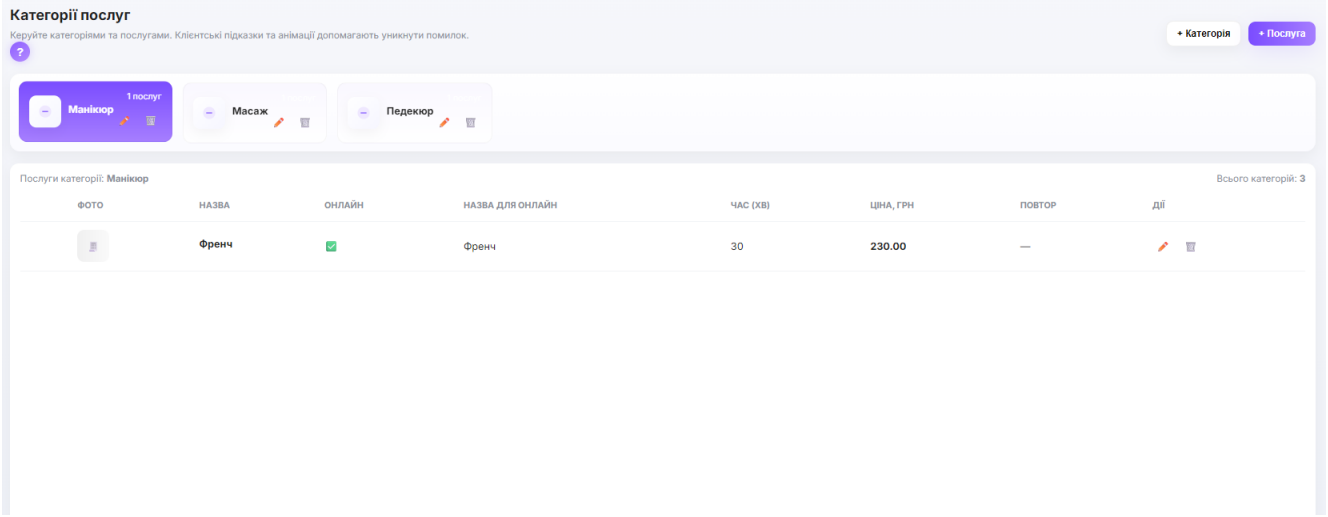


Рисунок 5.8 – Frot-end частина категорії та послуги

У результаті система реалізує повний функціонал керування категоріями та послугами, включаючи створення, редагування та видалення записів, із збереженням цілісності даних і динамічним інтерфейсом без перезавантаження сторінки.

Керування майстрами, яка охоплює створення, редагування та видалення записів у базі даних. Основна робота виконується через PHP-обробку POST-запитів, де кожна дія користувача напряму пов’язана з SQL-операціями над таблицями Users і Masters.

Створення нового майстра відбувається у блоці add_master рисунок 4.9, де спочатку з форми витягуються всі введені дані, включаючи логін, email, пароль, ім’я, прізвище, посаду, схему зарплати та прапорець можливості створення візитів. Далі виконується перевірка унікальності логіна або email через запит до таблиці Users. Якщо такий запис уже існує, виконання створення зупиняється і система повертає повідомлення про помилку. Якщо ж дані унікальні, спочатку створюється запис у таблиці Users через INSERT із використанням OUTPUT INSERTED.user_id, що дозволяє одразу отримати ідентифікатор нового користувача без додаткового запиту. Після цього цей user_id використовується для створення запису в таблиці Masters, де фіксується прив’язка майстра до конкретної філії через branch_id, а також зберігаються його професійні дані, включаючи посаду, ім’я, прізвище та службові права.

```php
/* ---------- POST ACTIONS (CRUD) ---------- */
if ($_SERVER['REQUEST_METHOD'] === 'POST') {

  // CREATE
  if (isset($_POST['add_master'])) {
    $fname = trim($_POST['first_name']);
    $lname = trim($_POST['last_name']);
    $pos = (int)$_POST['position_id'];
    $can_create = isset($_POST['can_create_visits']) ? 1 : 0;
    $salary = $_POST['salary_scheme_id'] ? (int)$_POST['salary_scheme_id'] : null;
    $login = trim($_POST['login']);
    $email = trim($_POST['email']);
    $password = $_POST['password'];

    // Проверка логина/почты
    $check = sqlsrv_query($conn, "SELECT user_id FROM Users WHERE login=? OR email=?", [$login, $email]);
    if ($check && sqlsrv_fetch_array($check, SQLSRV_FETCH_ASSOC)) {
      $message = "❌ Такий логін або email вже існує!";
    } else {
      $role_id = 2; // Master
      $pass_hash = $password;
      $created_at = date('Y-m-d H:i:s');

      $sqlInsert = "
        INSERT INTO Users (login, password_hash, email, phone, role_id, tariff_id, balance, created_at)
        OUTPUT INSERTED.user_id
        VALUES (?, ?, ?, NULL, ?, NULL, 0, ?)
      ";
      $stmt = sqlsrv_query($conn, $sqlInsert, [$login, $pass_hash, $email, $role_id, $created_at]);
      if ($stmt === false) { print_r(sqlsrv_errors()); exit; }

      $row = sqlsrv_fetch_array($stmt, SQLSRV_FETCH_ASSOC);
      $user_id_new = $row['user_id'];

      $insertMaster = sqlsrv_query($conn, "
        INSERT INTO Masters (user_id, branch_id, position_id, first_name, last_name, can_create_visits, salary_scheme_id)
        VALUES (?, ?, ?, ?, ?, ?, ?)",
        [$user_id_new, $branch_id, $pos, $fname, $lname, $can_create, $salary]
      );
      if (!$insertMaster) { print_r(sqlsrv_errors()); exit; }

      $message = "✅ Майстра створено разом із обліковим записом";
    }
  }
```

Рисунок 4.9 – Створення фахівця майстра

Редагування майстра реалізовано через окремий POST-блок edit_master рисунок 4.10, у якому виконується SQL UPDATE над таблицею Masters. Тут оновлюються основні поля запису, такі як ім’я, прізвище, посада, схема зарплати та можливість створення візитів. Важливо, що кожне оновлення виконується з додатковою умовою branch_id, що обмежує зміну даних лише в межах поточної філії і запобігає доступу до сторонніх записів.

Видалення майстра реалізовано через простий DELETE-запит до таблиці Masters, який також фільтрується за branch_id. Це забезпечує контроль цілісності даних і гарантує, що видалення можливе тільки для записів, які належать поточній філії.

```php
// UPDATE
if (isset($_POST['edit_master'])) {
  $mid = (int)$_POST['master_id'];
  $fname = trim($_POST['first_name']);
  $lname = trim($_POST['last_name']);
  $pos = (int)$_POST['position_id'];
  $salary = $_POST['salary_scheme_id'] ? (int)$_POST['salary_scheme_id'] : null;
  $can_create = isset($_POST['can_create_visits']) ? 1 : 0;

  $update = sqlsrv_query($conn, "
    UPDATE Masters
    SET first_name=?, last_name=?, position_id=?, salary_scheme_id=?, can_create_visits=?
    WHERE master_id=? AND branch_id=?",
    [$fname, $lname, $pos, $salary, $can_create, $mid, $branch_id]
  );

  if (!$update) { print_r(sqlsrv_errors()); exit; }
  $message = "🖍️ Дані майстра оновлено";
}

// DELETE
if (isset($_POST['delete_master'])) {
  $mid = (int)$_POST['master_id'];
  $del = sqlsrv_query($conn, "DELETE FROM Masters WHERE master_id=? AND branch_id=?", [$mid, $branch_id]);
  if (!$del) { print_r(sqlsrv_errors()); exit; }
  $message = "🗑️ Майстра видалено";
}

header("Location: " . $_SERVER['PHP_SELF'] . "?m=" . urlencode($message));
exit();
}

if (isset($_GET['m'])) $message = $_GET['m'];
```

Рисунок 4.10 – Редагування фахівця майстра

Після виконання будь-якої з операцій формується текстове повідомлення про результат, яке передається через GET-параметр і відображається на сторінці як індикатор успішності або помилки виконаної дії.

Окремий блок коду відповідає за завантаження даних для відображення інтерфейсу. Спочатку виконується вибірка посад із таблиці Positions, при цьому враховується як глобальні посади, так і ті, що прив’язані до конкретної філії. Далі завантажуються схеми зарплат із таблиці SalarySchemes. Основний список майстрів формується через SQL-запит з об’єднанням таблиць Masters, Positions і SalarySchemes, що дозволяє одразу отримати повний набір інформації про кожного майстра без необхідності додаткових запитів.

Отримані дані формуються у масив, де кожен елемент містить повний опис майстра, включаючи його особисті дані, посаду, схему зарплати та права доступу. Далі ці дані використовуються для побудови інтерфейсу сторінки, де кожен майстер відображається у вигляді окремої картки з основною інформацією та доступними діями.

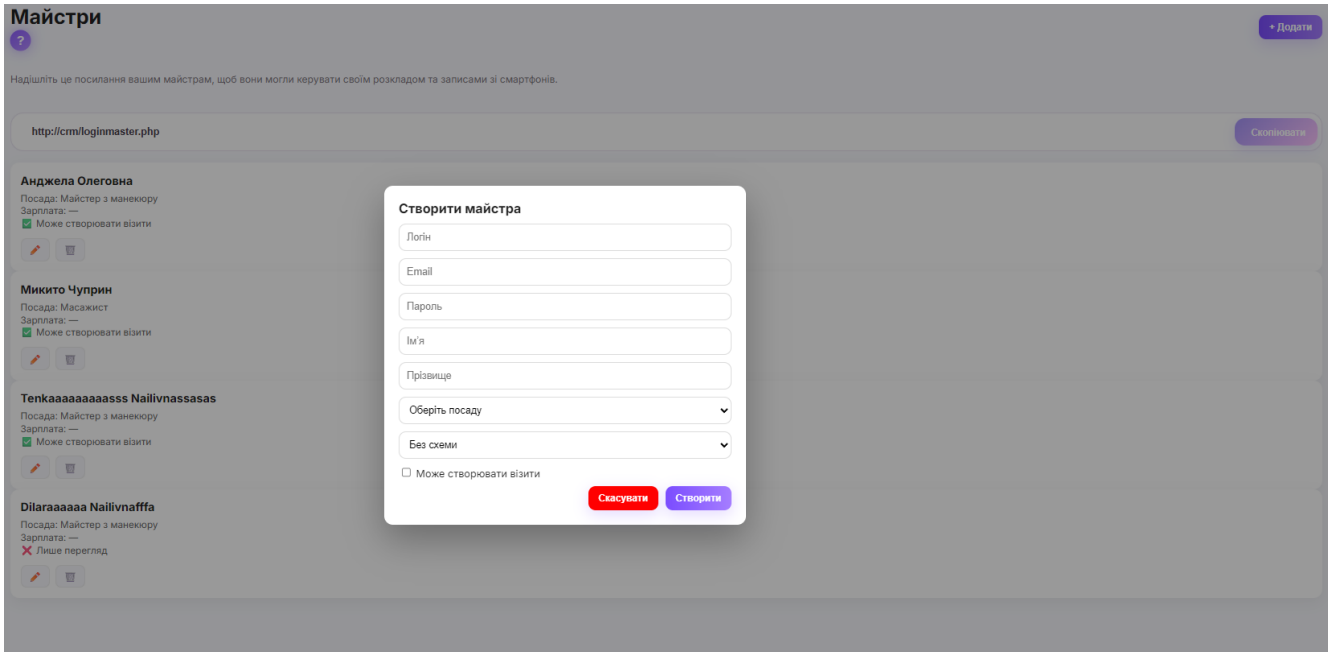


Рисунок 4.11 – Інтрефейс редагування майстрів

Інтерфейс редагування реалізовано через модальне вікно рисунок 4.11, яке заповнюється динамічно за допомогою JavaScript-функції openEditModal. Ця

функція приймає параметри обраного майстра і підставляє їх у форму редагування, після чого користувач може змінити значення і відправити їх назад на сервер для оновлення бази даних.

Повноцінний аналітичний модуль системи управління філією analiticr.php, де вся логіка побудована навколо збору, обробки та агрегування даних із бази SQL Server і подальшого представлення цих даних у вигляді дашборду. Основна ідея полягає в тому, що PHP тут не виконує складну бізнес-логіку, а виступає як шар, який формує запити до бази даних, отримує вже оброблені результати та передає їх у фронтенд для візуалізації.

На початку відбувається визначення користувача через email із сесії. Це потрібно не для додаткових перевірок, а для того, щоб отримати user_id, який далі використовується як ключ доступу до конкретної філії. Після цього через таблицю Branches визначається branch_id, і саме цей ідентифікатор стає основним фільтром для всіх наступних SQL-запитів. Таким чином система жорстко ізолює дані між різними філіями і гарантує, що кожен власник бачить тільки свою аналітику.

Далі починається блок збору ключових метрик. Загальна кількість візитів отримується через простий агрегаційний запит COUNT(*) з таблиці Visits, де використовується фільтр по branch_id.

Це базовий показник активності системи. Паралельно виконується окремий запит для завершених візитів через умову status = 'completed' рисунок 4.12, а також для скасованих через status = 'cancelled'. Таким чином формується повна структура станів бізнес-процесу: скільки записів створено, скільки виконано і скільки втрачено.

```php
/* ================ VISITS ================ */
$sql = "SELECT COUNT(*) AS cnt FROM Visits WHERE branch_id = ?";
$stmt = sqlsrv_query($conn, $sql, [$branch_id]);
$visits = sqlsrv_fetch_array($stmt, SQLSRV_FETCH_ASSOC)['cnt'] ?? 0;

$sql = "SELECT COUNT(*) AS cnt FROM Visits WHERE branch_id = ? AND status = 'completed'";
$stmt = sqlsrv_query($conn, $sql, [$branch_id]);
$done_visits = sqlsrv_fetch_array($stmt, SQLSRV_FETCH_ASSOC)['cnt'] ?? 0;

$sql = "SELECT COUNT(*) AS cnt FROM Visits WHERE branch_id = ? AND status = 'cancelled'";
$stmt = sqlsrv_query($conn, $sql, [$branch_id]);
$cancelled = sqlsrv_fetch_array($stmt, SQLSRV_FETCH_ASSOC)['cnt'] ?? 0;

/* ================ CLIENTS ================ */
$sql = "SELECT COUNT(*) AS cnt FROM Clients WHERE branch_id = ?";
$stmt = sqlsrv_query($conn, $sql, [$branch_id]);
$clients = sqlsrv_fetch_array($stmt, SQLSRV_FETCH_ASSOC)['cnt'] ?? 0;

/* ================ DEBT ================ */
$sql = "
SELECT ISNULL(SUM(me.amount),0) AS debt
FROM MasterEarnings me
JOIN Masters m ON m.master_id = me.master_id
WHERE m.branch_id = ? AND me.paid = 0
";
$stmt = sqlsrv_query($conn, $sql, [$branch_id]);
$debt = sqlsrv_fetch_array($stmt, SQLSRV_FETCH_ASSOC)['debt'] ?? 0;

/* ================ TOTAL INCOME ================ */
$sql = "
SELECT ISNULL(SUM(change),0) AS total
FROM CashRegister
WHERE branch_id = ? AND change > 0
";
$stmt = sqlsrv_query($conn, $sql, [$branch_id]);
$total_income = sqlsrv_fetch_array($stmt, SQLSRV_FETCH_ASSOC)['total'] ?? 0;
```

Рисунок 4.12 – Аналітика CRM системи

Окремим блоком рахується кількість клієнтів через таблицю Clients, також з фільтром по філії. Це дозволяє оцінити масштаб бази клієнтів і співвіднести його з кількістю візитів. Далі йде фінансовий блок, де використовується таблиця CashRegister. Поле change виступає як універсальний показник грошового руху, і через SUM(change) отримується загальний дохід філії. Це класична схема бухгалтерської агрегації, де всі надходження просто підсумовуються.

Більш складною частиною є періодична аналітика доходів. Тут використовується динамічний SQL-фільтр $dateFilter, який змінюється залежно від параметра period. У коді він може перетворюватися на умови типу “сьогодні”, “останні 7 днів”, “30 днів” або “рік”. Це дозволяє одному і тому ж запиту працювати в різних часових режимах без дублювання логіки. Таким чином система отримує гнучку модель аналітики, де користувач сам керує періодом аналізу.

Наступний блок це розрахунок заборгованості по майстрах. Тут використовується таблиця MasterEarnings, де відбираються тільки ті записи, які ще не позначені як оплачені (paid = 0). Після цього виконується сумування через SUM(amount), що дає загальний борг перед співробітниками. Це дозволяє контролювати фінансові зобов’язання системи в реальному часі.

Середній чек розраховується через AVG(total_amount) у таблиці Visits, але тільки для завершених візитів. Це важливо, тому що система навмисно ігнорує незавершені або некоректні записи, щоб не спотворювати статистику. Додатково рахується кількість нових клієнтів за останні 30 днів через умову created_at >= DATEADD(DAY, -30, GETDATE()), що дозволяє оцінити динаміку росту клієнтської бази.

Окремо формується показник конверсії, який обчислюється вже на рівні PHP як відношення завершених візитів до загальної кількості. Це простий, але важливий бізнес-індикатор, який показує ефективність роботи філії.

Далі формується блок даних для графіків. Тут SQL групує записи по даті через CAST(created_at AS DATE) і сумує доходи за кожен день. У результаті формується часовий ряд, який потім перетворюється в два масиви: список дат і список значень доходу. Ці масиви передаються у Chart.js, де вже на фронтенді будується лінійний графік. Таким чином PHP відповідає за підготовку даних, а JavaScript — за їх візуальне представлення.

Рейтинг майстрів формується через об’єднання таблиць MasterEarnings і Masters. Тут дані групуються по кожному майстру і сумуються через SUM(amount), після чого результат сортується за спаданням. Це дозволяє отримати топ

найприбутковіших співробітників філії. Подібний підхід часто використовується в CRM-системах для мотиваційної аналітики персоналу.

Окремо формується розподіл статусів візитів через GROUP BY status, що дозволяє побудувати кругову діаграму. Це дає швидке візуальне розуміння того, яка частина записів виконана, скасована або знаходиться в очікуванні.

Найскладніший блок це деталізація нарахувань. Тут SQL-запит об’єднує одразу кілька таблиць: MasterEarnings, Masters, Visits і Services. Це дозволяє отримати повну фінансову транзакцію: який майстер отримав дохід, за яку послугу, яка її вартість і який відсоток був нарахований. Відсоток рахується прямо в SQL через формулу (me.amount / v.total_amount) * 100, що знімає необхідність додаткових обчислень у PHP. У результаті система одразу повертає готові бізнес-показники, а не сирі дані.

У фінальному результаті весь код працює як аналітичний дашборд рівня CRM-системи: SQL Server виконує всю важку роботу по агрегації даних, PHP виступає як проміжний шар збору і підготовки, а фронтенд (Chart.js і HTML-таблиці) відповідає за відображення.

Модуль онлайн-запису рисунок 4.13 реалізований у вигляді окремого веб-віджета, який дозволяє клієнту самостійно створювати запис на послугу без авторизації в системі. Основною метою даного модуля є автоматизація процесу бронювання часу та зменшення навантаження на адміністраторів салону.

Інтерфейс побудований на основі HTML, CSS та JavaScript і працює за принципом послідовного вибору параметрів візиту. Користувач спочатку обирає філію підприємства. Після вибору виконується асинхронний запит до файлу booking_api.php з параметром get_branches, який звертається до таблиці Branches та повертає список усіх доступних філій разом із їх адресами. Отримані дані автоматично заповнюють випадаючий список на сторінці.

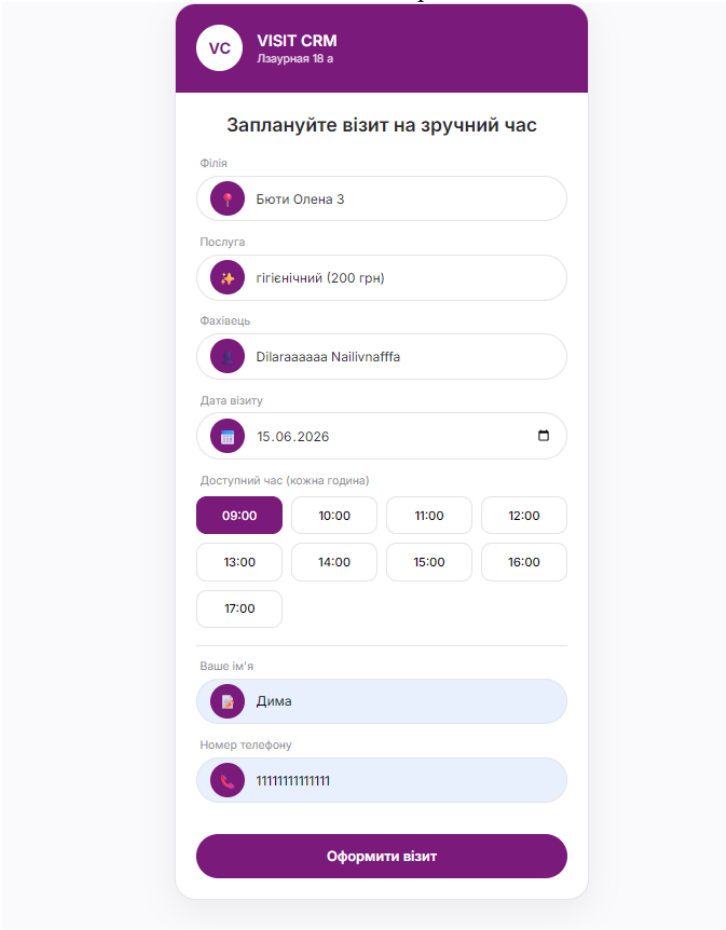


Рисунок 4.13 – Віджет онлайн запису CRM системи

Після вибору філії система виконує запит get_services. На серверній стороні відбувається вибірка послуг, які доступні саме у вибраній філії. Для цього використовується зв'язок між таблицями Branches, Services та додатковими таблицями налаштування послуг. У результаті користувач бачить лише ті послуги, які реально можуть бути надані у вибраному відділенні. Одночасно система відображає вартість кожної послуги, що дозволяє клієнту одразу оцінити майбутні витрати.

Наступним етапом є вибір майстра. Після вибору послуги клієнт ініціює запит get_masters, який звертається до таблиці Masters. Для визначення доступних спеціалістів використовується таблиця PositionServices, яка містить інформацію про те, які послуги може виконувати конкретний співробітник. Додатково враховується прив'язка майстра до певної філії. Завдяки цьому клієнт бачить лише тих працівників, які мають необхідну кваліфікацію для виконання обраної послуги.

Після вибору майстра активується поле вибору дати візиту. Для запобігання помилкам система забороняє вибір минулих дат шляхом автоматичного встановлення мінімального значення календаря на поточний день. Після вибору дати виконується запит get_time_slots, який формує список доступних часових інтервалів.

При побудові списку вільного часу серверна частина аналізує таблицю Visits, де зберігаються всі створені записи клієнтів. Система перевіряє наявність уже заброньованих візитів для вибраного майстра на зазначену дату та виключає зайняті часові проміжки із загального розкладу. У результаті клієнту відображаються лише вільні години для запису. Якщо на обрану дату вільних слотів немає, система генерує спеціальне повідомлення про відсутність доступного часу та пропонує обрати іншу дату.

Після вибору часу відображається форма введення персональних даних клієнта. Користувач вказує своє ім'я та номер телефону. Ця інформація використовується для створення нового запису та подальшого зв'язку з клієнтом у разі необхідності підтвердження або перенесення візиту.

Під час натискання кнопки оформлення запису всі вибрані дані передаються методом POST до серверного обробника create_booking. На сервері виконується додаткова перевірка коректності отриманих даних та перевірка доступності вибраного часу. Після успішної валідації створюється новий запис у таблиці Visits, де зберігаються ідентифікатори філії, послуги, майстра, дата та час візиту, а також контактні дані клієнта.

Під час створення нового візиту система також може автоматично додавати нового клієнта до таблиці Clients, якщо вказаний номер телефону раніше не зустрічався в базі даних. Таким чином реалізується автоматичне накопичення клієнтської бази без участі адміністратора.

Архітектура модуля побудована за клієнт-серверним принципом. JavaScript відповідає за взаємодію користувача з інтерфейсом та надсилання AJAX-запитів. PHP виконує бізнес-логіку, перевірку даних і роботу з базою даних SQL Server.

База даних забезпечує зберігання інформації про філії, послуги, майстрів, клієнтів та візити. Такий підхід дозволяє створити повністю автоматизований механізм онлайн-бронювання, який працює в режимі реального часу та виключає можливість подвійного резервування одного й того самого часу.

Серверна частина модуля онлайн-запису реалізована у вигляді API-контролера booking_api.php рисунок 4.14, який працює за принципом REST-подібної взаємодії між клієнтською частиною та базою даних. Всі операції виконуються через параметр action, який визначає необхідний сценарій обробки запиту. Після отримання запиту відбувається підключення до бази даних SQL Server через клас NewsDB, який відповідає за створення та підтримку з'єднання.

Першим функціональним блоком є отримання списку філій. Для цього використовується запит до таблиці Branches, з якої вибираються поля branch_id, name та address. Отримані записи перетворюються у формат JSON і передаються клієнтській частині. Таким чином система формує початковий список доступних філій, з яких користувач може обрати потрібне відділення.

Наступним етапом є формування списку послуг. На відміну від простого виведення всіх послуг, система виконує додаткову бізнес-перевірку. SQL-запит об'єднує таблиці Services, PositionServices та Masters. Завдяки цьому повертаються лише ті послуги, які можуть бути виконані працівниками конкретної філії. Такий підхід забезпечує цілісність даних і виключає ситуації, коли клієнт може записатися на послугу, для якої немає відповідного спеціаліста.

Після вибору послуги виконується пошук доступних майстрів. Для цього використовується зв'язок між таблицями Masters та PositionServices. Кожен майстер має певну посаду, а кожна посада містить перелік доступних послуг. У результаті система автоматично визначає перелік співробітників, які можуть виконувати обрану процедуру у конкретній філії. Отримані дані повертаються у вигляді списку ідентифікаторів та прізвищ працівників.

```php
if ($action === 'get_services') {
    $branch_id = intval($_GET['branch_id'] ?? 0);

    if ($branch_id > 0) {

        $sql = "SELECT DISTINCT s.service_id, s.name, s.price, s.duration_minutes
                FROM Services s
                JOIN PositionServices ps ON s.service_id = ps.service_id
                JOIN Masters m ON m.position_id = ps.position_id
                WHERE m.branch_id = ?
                ORDER BY s.name ASC";
        $stmt = sqlsrv_query($conn, $sql, [$branch_id]);
    } else {

        $stmt = sqlsrv_query($conn, "SELECT service_id, name, price, duration_minutes FROM Services ORDER BY name ASC");
    }

    $services = [];
    if ($stmt !== false) {
        while ($row = sqlsrv_fetch_array($stmt, SQLSRV_FETCH_ASSOC)) {
            $services[] = $row;
        }
    }

    echo json_encode($services);
    exit;
}


if ($action === 'get_masters') {
    $service_id = intval($_GET['service_id'] ?? 0);
    $branch_id = intval($_GET['branch_id'] ?? 0);


    $sql = "SELECT m.master_id, m.first_name, m.last_name
            FROM Masters m
            JOIN PositionServices ps ON m.position_id = ps.position_id
            WHERE ps.service_id = ? AND m.branch_id = ?";

    $stmt = sqlsrv_query($conn, $sql, [$service_id, $branch_id]);
    $masters = [];
    while ($row = sqlsrv_fetch_array($stmt, SQLSRV_FETCH_ASSOC)) {
        $masters[] = $row;
    }
    echo json_encode($masters);
    exit;
}


if ($action === 'get_time_slots') {
    $master_id = intval($_GET['master_id'] ?? 0);
    $date = $_GET['date'] ?? ''; // Формат YYYY-MM-DD

    if (!$master_id || !$date) {
        echo json_encode([]);
        exit;
    }


    $day_of_week = date('N', strtotime($date));


    $sql_schedule = "SELECT start_time, end_time, is_working FROM MasterSchedule WHERE master_id = ? AND day_of_week = ?";
    $stmt_sched = sqlsrv_query($conn, $sql_schedule, [$master_id, $day_of_week]);
    $sched = sqlsrv_fetch_array($stmt_sched, SQLSRV_FETCH_ASSOC);


    if (!$sched || !$sched['is_working']) {
        echo json_encode([]);
        exit;
    }


    $start_hour = intval($sched['start_time'] instanceof DateTime ? $sched['start_time']->format('H') : substr($sched['start_time'], 0, 2));
    $end_hour = intval($sched['end_time'] instanceof DateTime ? $sched['end_time']->format('H') : substr($sched['end_time'], 0, 2));


$sql_visits = "
SELECT scheduled_start, scheduled_end
FROM Visits
WHERE master_id = ?
AND CAST(scheduled_start AS DATE) = ?
AND status != 'cancelled'
";
```

Рисунок 4.14 – Фрагмент коду серверної части онлайн запису CRM системи

Найбільш складним компонентом API є механізм генерації вільних часових слотів. Після отримання дати та майстра система спочатку визначає день тижня за допомогою функцій роботи з датою. Далі виконується звернення до таблиці MasterSchedule, яка містить індивідуальний графік роботи кожного працівника. З таблиці отримуються час початку зміни, час завершення роботи та ознака робочого

дня. Якщо працівник у цей день не працює або графік відсутній, система повертає порожній список доступного часу.

Після визначення робочого інтервалу виконується аналіз уже створених записів у таблиці Visits. Для вибраного майстра та дати вибираються всі активні візити, статус яких не дорівнює cancelled. Із записів отримуються значення scheduled_start та scheduled_end, які визначають фактичний проміжок часу зайнятості працівника. Далі алгоритм проходить кожну годину всередині цього інтервалу та формує перелік заблокованих часових слотів. Такий механізм дозволяє враховувати не лише початок візиту, а й повну його тривалість.

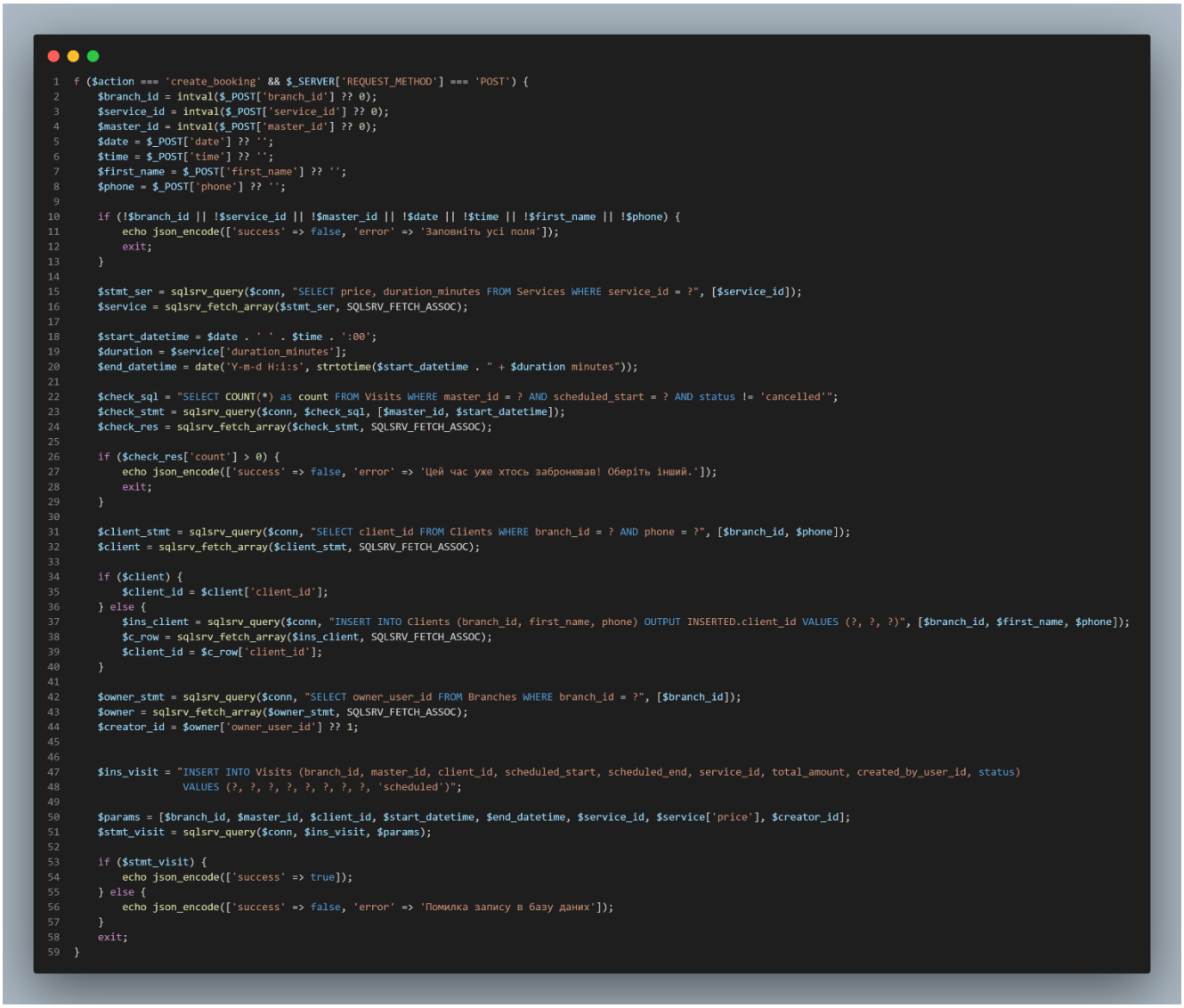
На основі робочого графіка та зайнятих інтервалів формується остаточний список доступного часу. Система генерує часові слоти з інтервалом одна година та виключає всі значення, які вже використовуються іншими записами. У результаті клієнт отримує тільки ті часові проміжки, які реально доступні для бронювання.

Окремий блок відповідає за створення нового візиту. Після отримання даних від користувача система виконує перевірку наявності всіх обов'язкових параметрів. Далі відбувається звернення до таблиці Services для отримання вартості послуги та її тривалості. На основі вибраного часу початку автоматично розраховується час завершення візиту, що дозволяє коректно резервувати графік майстра.

Перед створенням запису виконується додаткова перевірка конкурентного доступу. Система звертається до таблиці Visits і перевіряє, чи не був вибраний часовий слот заброньований іншим клієнтом між моментом завантаження сторінки та натисканням кнопки підтвердження. Такий механізм захищає систему від подвійного бронювання одного й того самого часу.

Після проходження перевірки виконується пошук клієнта в таблиці Clients. Пошук здійснюється за номером телефону та ідентифікатором філії. Якщо клієнт уже існує, використовується його поточний ідентифікатор. Якщо запис відсутній, система автоматично створює нового клієнта та отримує його первинний ключ через конструкцію OUTPUT INSERTED.client_id. Завдяки цьому клієнтська база наповнюється автоматично без додаткових дій адміністратора.

Далі виконується визначення користувача, від імені якого створюється запис. Для цього система звертається до таблиці Branches та отримує значення owner_user_id. Даний ідентифікатор використовується для заповнення поля created_by_user_id у журналі візитів, що забезпечує можливість подальшого аудиту та контролю дій у системі.



```
f ($action === 'create_booking' && $_SERVER['REQUEST_METHOD'] === 'POST') {
    $branch_id = intval($_POST['branch_id'] ?? 0);
    $service_id = intval($_POST['service_id'] ?? 0);
    $master_id = intval($_POST['master_id'] ?? 0);
    $date = $_POST['date'] ?? '';
    $time = $_POST['time'] ?? '';
    $first_name = $_POST['first_name'] ?? '';
    $phone = $_POST['phone'] ?? '';

    if (!$branch_id || !$service_id || !$master_id || !$date || !$time || !$first_name || !$phone) {
        echo json_encode(['success' => false, 'error' => 'Заповніть усі поля']);
        exit;
    }

    $stmt_ser = sqlsrv_query($conn, "SELECT price, duration_minutes FROM Services WHERE service_id = ?", [$service_id]);
    $service = sqlsrv_fetch_array($stmt_ser, SQLSRV_FETCH_ASSOC);

    $start_datetime = $date . ' ' . $time . ':00';
    $duration = $service['duration_minutes'];
    $end_datetime = date('Y-m-d H:i:s', strtotime($start_datetime . " + $duration minutes"));

    $check_sql = "SELECT COUNT(*) as count FROM Visits WHERE master_id = ? AND scheduled_start = ? AND status != 'cancelled'";
    $check_stmt = sqlsrv_query($conn, $check_sql, [$master_id, $start_datetime]);
    $check_res = sqlsrv_fetch_array($check_stmt, SQLSRV_FETCH_ASSOC);

    if ($check_res['count'] > 0) {
        echo json_encode(['success' => false, 'error' => 'Цей час уже хтось забронював! Оберіть інший.']);
        exit;
    }

    $client_stmt = sqlsrv_query($conn, "SELECT client_id FROM Clients WHERE branch_id = ? AND phone = ?", [$branch_id, $phone]);
    $client = sqlsrv_fetch_array($client_stmt, SQLSRV_FETCH_ASSOC);

    if ($client) {
        $client_id = $client['client_id'];
    } else {
        $ins_client = sqlsrv_query($conn, "INSERT INTO Clients (branch_id, first_name, phone) OUTPUT INSERTED.client_id VALUES (?, ?, ?)", [$branch_id, $first_name, $phone]);
        $c_row = sqlsrv_fetch_array($ins_client, SQLSRV_FETCH_ASSOC);
        $client_id = $c_row['client_id'];
    }

    $owner_stmt = sqlsrv_query($conn, "SELECT owner_user_id FROM Branches WHERE branch_id = ?", [$branch_id]);
    $owner = sqlsrv_fetch_array($owner_stmt, SQLSRV_FETCH_ASSOC);
    $creator_id = $owner['owner_user_id'] ?? 1;


    $ins_visit = "INSERT INTO Visits (branch_id, master_id, client_id, scheduled_start, scheduled_end, service_id, total_amount, created_by_user_id, status)
                  VALUES (?, ?, ?, ?, ?, ?, ?, ?, 'scheduled')";

    $params = [$branch_id, $master_id, $client_id, $start_datetime, $end_datetime, $service_id, $service['price'], $creator_id];
    $stmt_visit = sqlsrv_query($conn, $ins_visit, $params);

    if ($stmt_visit) {
        echo json_encode(['success' => true]);
    } else {
        echo json_encode(['success' => false, 'error' => 'Помилка запису в базу даних']);
    }
    exit;
}
```

Рисунок 4.15 – Фрагмент коду створення візиту через онлайн запису CRM системи

Завершальним етапом є створення нового запису рисунок 4.15 у таблиці Visits. У запис зберігаються ідентифікатор філії, майстра, клієнта, послуги, дата початку, дата завершення, вартість послуги, користувач-ініціатор та початковий статус scheduled. Після успішного виконання операції API повертає клієнтській

частині JSON-відповідь із ознакою success, що сигналізує про успішне оформлення бронювання.

Таким чином серверний модуль онлайн-запису реалізує повний цикл обробки заявки: від отримання довідкової інформації про філії та послуги до перевірки доступності часу, автоматичного створення клієнта та формування нового візиту в базі даних. Основні обчислення та перевірки виконуються на стороні SQL Server і PHP, що забезпечує високу надійність роботи системи та цілісність даних.

Висновки до розділу 4

У четвертому розділі було виконано розробку та програмну реалізацію CRM-системи управління візитами для підприємств сфери послуг. На основі проведеного проєктування створено реляційну базу даних у середовищі Microsoft SQL Server, яка забезпечує зберігання та обробку інформації про користувачів, філії, майстрів, клієнтів, послуги, візити та фінансові операції. Реалізована структура бази даних відповідає принципам нормалізації, що дозволило мінімізувати дублювання даних та забезпечити їх цілісність за допомогою первинних і зовнішніх ключів.

У процесі реалізації серверної частини системи використано мову програмування PHP та драйвер SQLSRV для взаємодії з базою даних. Було розроблено модулі реєстрації та авторизації користувачів, керування філіями, майстрами, категоріями та послугами, а також механізм створення, редагування та обробки візитів. Особливу увагу приділено автоматизації фінансового обліку, де реалізовано механізми нарахування заробітної плати майстрам, ведення касових операцій та контролю фінансових транзакцій.

Також було реалізовано аналітичний модуль, який забезпечує збір та обробку статистичних даних щодо діяльності філії. Система дозволяє отримувати інформацію про кількість візитів, клієнтів, дохід підприємства, середній чек, заборгованість перед працівниками та інші показники ефективності. Для візуального представлення результатів використано інтерактивні графіки та діаграми, що підвищує зручність аналізу даних користувачем.

Окремим напрямом реалізації став модуль онлайн-запису клієнтів. Розроблений веб-віджет забезпечує можливість самостійного бронювання послуг без необхідності авторизації в системі. Реалізований механізм автоматично формує доступні часові слоти на основі графіків роботи майстрів та вже існуючих записів, що дозволяє уникнути конфліктів розкладу та подвійного бронювання. Серверна частина модуля забезпечує перевірку коректності даних, автоматичне створення клієнтів у базі даних та формування нових записів про візити.

У результаті виконаних робіт створено повноцінну CRM-систему, яка забезпечує автоматизацію основних бізнес-процесів підприємств сфери послуг, підвищує ефективність управління клієнтською базою, персоналом та фінансами, а також надає інструменти для оперативного аналізу діяльності підприємства. Реалізоване програмне забезпечення відповідає поставленим вимогам та може бути використане як основа для подальшого розвитку і впровадження в реальних умовах експлуатації.

ВИСНОВКИ

У результаті виконання дипломної роботи було проведено дослідження сучасних підходів до автоматизації діяльності підприємств сфери послуг та розроблено CRM-систему управління клієнтами, візитами, персоналом і фінансовими процесами. Актуальність роботи обумовлена необхідністю цифровізації бізнес-процесів, підвищення ефективності управління ресурсами підприємства та забезпечення якісного обслуговування клієнтів в умовах зростання конкуренції на ринку послуг.

У теоретичній частині роботи було проаналізовано особливості функціонування CRM-систем, їх роль у діяльності сучасних підприємств та основні принципи організації взаємодії з клієнтами. Проведено аналіз існуючих програмних рішень, визначено їх переваги та недоліки, а також сформовано перелік функціональних вимог до майбутньої системи. На основі проведеного аналізу було встановлено, що існуючі програмні продукти часто є надмірно складними для невеликих підприємств або потребують значних фінансових витрат на впровадження та підтримку. Саме тому виникла необхідність створення власного програмного рішення, орієнтованого на специфіку роботи підприємств сфери послуг.

У процесі проєктування було визначено архітектуру системи, побудовано модель бази даних та розроблено структуру взаємодії між основними сутностями. Для зберігання інформації обрано реляційну систему керування базами даних Microsoft SQL Server. Розроблена база даних містить таблиці користувачів, ролей, філій, клієнтів, майстрів, послуг, категорій послуг, візитів, графіків роботи, касових операцій та фінансових нарахувань. Використання механізму первинних та зовнішніх ключів забезпечило цілісність даних і правильність взаємозв'язків між об'єктами системи.

У рамках практичної реалізації було створено веборієнтовану CRM-систему з використанням мови програмування PHP, технологій HTML, CSS та JavaScript. Для взаємодії з базою даних використано драйвер SQLSRV, що забезпечує

надійний обмін інформацією між прикладним програмним забезпеченням та SQL Server. Реалізовано систему автентифікації та авторизації користувачів із розмежуванням прав доступу відповідно до ролей. Кожен тип користувачів отримує доступ лише до функціоналу, необхідного для виконання його посадових обов'язків.

У розробленій системі реалізовано повний цикл роботи з клієнтами. Передбачено можливість створення, редагування та пошуку клієнтів, перегляду історії відвідувань, ведення контактної інформації та аналізу взаємодії з клієнтською базою. Автоматизація цих процесів дозволяє значно скоротити час на обробку інформації та підвищити якість обслуговування.

Одним із ключових модулів системи став механізм управління візитами. Реалізовано створення, редагування, перенесення та скасування записів клієнтів. Система автоматично контролює графіки роботи майстрів, запобігає виникненню конфліктів розкладу та виключає можливість подвійного бронювання часу. Це забезпечує ефективне використання робочого часу персоналу та підвищує якість організації діяльності підприємства.

Важливою складовою розробленого програмного продукту став модуль онлайн-запису. Реалізований вебвіджет дозволяє клієнтам самостійно обирати філію, послугу, спеціаліста та зручний час відвідування. Система автоматично аналізує графіки роботи майстрів і вже створені записи, після чого формує перелік доступних часових інтервалів. У результаті клієнт може оформити запис без участі адміністратора, а інформація про візит миттєво потрапляє до бази даних системи.

Особливу увагу приділено фінансовому обліку та аналітиці. У системі реалізовано механізми обліку доходів підприємства, нарахування заробітної плати майстрам, контролю касових операцій та відстеження фінансових показників діяльності. Передбачено формування статистичних звітів щодо кількості клієнтів, візитів, середнього чека, доходів підприємства та інших показників ефективності. Для наочного представлення інформації використовуються графіки та діаграми, що значно спрощує процес аналізу даних і прийняття управлінських рішень.

У результаті виконання роботи було досягнуто поставленої мети та виконано всі визначені завдання. Створена CRM-система забезпечує автоматизацію основних бізнес-процесів підприємств сфери послуг, сприяє підвищенню продуктивності праці персоналу, зменшенню кількості помилок під час обробки інформації та покращенню якості обслуговування клієнтів. Використання єдиної інформаційної системи дозволяє централізовано керувати всіма процесами підприємства та оперативно отримувати необхідну управлінську інформацію.

Практична цінність розробленого програмного забезпечення полягає в можливості його використання в реальних умовах функціонування підприємств сфери послуг, зокрема салонів краси, барбершопів, косметологічних центрів та інших організацій, діяльність яких пов'язана з попереднім записом клієнтів. Архітектура системи дозволяє розширювати її функціональні можливості шляхом додавання нових модулів без необхідності суттєвої зміни існуючої структури.

Перспективами подальшого розвитку проєкту можуть стати інтеграція із зовнішніми платіжними системами, реалізація SMS та електронних повідомлень для нагадування про візити, впровадження мобільного застосунку для клієнтів і співробітників, а також використання інструментів штучного інтелекту для прогнозування навантаження персоналу та аналізу поведінки клієнтів. Це дозволить підвищити ефективність функціонування системи та розширити сферу її практичного застосування.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Sommerville I. Software Engineering. 10th ed. Pearson, 2016. 816 p.
2. Pressman R. S., Maxim B. R. Software Engineering: A Practitioner's Approach. 8th ed. McGraw-Hill, 2019. 970 p.
3. Newman S. Building Microservices. 2nd ed. O'Reilly Media, 2021. 617 p.
4. Fowler M. Microservices: A Definition of This New Architectural Term. URL: https://martinfowler.com/articles/microservices.html (дата звернення: 04.04.2026).
5. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1994. 395 p.
6. ISO/IEC 25010:2011. Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE).
7. ISO/IEC/IEEE 29148:2018. Systems and software engineering - Life cycle processes - Requirements engineering.
8. Wieringa R. Design Science Methodology for Information Systems and Software Engineering. Springer, 2014. 493 p.
9. Kumar V., Reinartz W. Customer Relationship Management: Concept, Strategy, and Tools. Springer, 2018. 350 p.
10. Chen I. J., Popovich K. Understanding customer relationship management (CRM) // Business Process Management Journal. 2003. Vol. 9, No. 5. P. 672-688.
11. PHP: Hypertext Preprocessor - Official Manual. URL: https://www.php.net/manual/en/ (дата звернення: 04.04.2026).
12. Welling L., Thomson L. PHP and MySQL Web Development. Addison-Wesley, 2016.
13. Ullman L. PHP and MySQL for Dynamic Web Sites. Peachpit Press, 2019.
14. MySQL 8.0 Reference Manual. Oracle. URL: https://dev.mysql.com/doc/refman/8.0/en/ (дата звернення: 04.04.2026).
15. Microsoft SQL Server 2019 Documentation. Microsoft Docs. URL: https://learn.microsoft.com/en-us/sql/sql-server/ (дата звернення: 04.04.2026).

16. Open Server documentation. URL: https://ospanel.io/documentation/ (дата звернення: 04.04.2026).
17. Duckett J. HTML and CSS: Design and Build Websites. Wiley. URL: https://www.wiley.com/en-us/HTML+and+CSS%3A+Design+and+Build+Websites-p-9781118008188 (дата звернення: 04.04.2026).
18. HTML Living Standard (WHATWG). URL: https://html.spec.whatwg.org/ (дата звернення: 04.04.2026).
19. Mozilla Developer Network - HTML documentation. URL: https://developer.mozilla.org/en-US/docs/Web/HTML (дата звернення: 04.04.2026).
20. Mozilla Developer Network - CSS documentation. URL: https://developer.mozilla.org/en-US/docs/Web/CSS (дата звернення: 04.04.2026).
21. Flanagan D. JavaScript: The Definitive Guide. O'Reilly Media, 2020. URL: https://www.oreilly.com/library/view/javascript-the-definitive/9781491952016/ (дата звернення: 04.04.2026).
22. Mozilla Developer Network - JavaScript Guide. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide (дата звернення: 04.04.2026).
23. Zakas N. C. Professional JavaScript for Web Developers. Wrox, 2011.
24. CSS: Cascading Style Sheets - W3C Recommendation. URL: https://www.w3.org/Style/CSS/ (дата звернення: 04.04.2026).
25. ECMAScript® Language Specification. ECMA International. URL: https://262.ecma-international.org/ (дата звернення: 04.04.2026).
26. Borland A. Learning PHP 7 High Performance. Packt Publishing, 2016.
27. SQL Server Tutorials - Microsoft Learn. URL: https://learn.microsoft.com/en-us/sql/relational-databases/tutorials/ (дата звернення: 04.04.2026).
28. PHP The Right Way - сайт-гайд по «правильному PHP». URL: https://phptherightway.com/ (дата звернення: 04.04.2026).

29. Stack Overflow Developer Survey 2025 - сучасні тренди веб-розробки. URL: https://stackoverflow.com/survey/2025 (дата звернення: 04.04.2026).
30. OWASP - Top 10 Web Application Security Risks - безпека PHP/JS. URL: https://owasp.org/www-project-top-ten/ (дата звернення: 04.04.2026).