НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ

Завідувач кафедри

комп'ютерних наук
(назва кафедри)

_____ Голуб Б.Л.
(підпис) (ПІБ)

"_04_" ___червня___ 2025 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему

«Інформаційна система управління медичними записами»

Спеціальність 122 – «Комп'ютерні науки»

Гарант освітньої програми

_____Д.е.н., професор_____ _____ ___Руденський Р.А.____
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник бакалаврської кваліфікаційної роботи

_____ _____ ___Панкратьєв В.О._____
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконала _____ _____Музиченко А.А._____
(підпис) (ПІБстудента)

КИЇВ – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри
комп'ютерних наук
к.т.н., доцент _____ Голуб Б.Л.
(науковий ступінь, вчене звання) (підпис) (ПІБ)
"16"грудня 2024 р.

З А В Д А Н Н Я
на виконання бакалаврської кваліфікаційної роботи студентці
Музиченко Анастасії Анатоліївни
(прізвище, ім'я, по батькові)

Спеціальність 122 – «Комп'ютерні науки»

Тема бакалаврської кваліфікаційної роботи «Інформаційна система управління медичними записами»
затверджена наказом ректора НУБіП України №2246 С від "16" 12 2024р.

Термін подання завершеної роботи на кафедру 2025. 05. 25
(рік, місяць, число)

Вихідні дані до бакалаврської кваліфікаційної роботи
Публічні джерела,

Перелік питань, які потрібно розробити:
Системний аналіз предметної області, інформаційне забезпечення, прикладне програмне забезпечення, рекомендації щодо впровадження та експлуатації системи

Перелік графічних документів (за потреби) діаграми та моделі

Керівник бакалаврської кваліфікаційної роботи _____ Панкратьєв В.О.
(підпис) (прізвище та ініціали)

Завдання прийняла до виконання _____ Музиченко А.А.
(підпис) (прізвище та ініціали студента)

Дата отримання завдання "16" грудня 2024р.

ЗМІСТ

Перелік умовних позначень

СУБД – система управління базами даних.

ІС – інформаційна система.

БД – база даних.

ER – Entity–Relationship (сутність–зв’язок).

ППЗ – прикладне програмне забезпечення.

UML – Unified Modeling Language (уніфікована мова моделювання).

RAM – Random Access Memory (оперативна пам’ять).

IDE – Integrated Development Environment (інтегроване середовище розробки).

ВСТУП

Сучасний світ інформаційних технологій характеризується активною інтеграцією цифрових рішень у всі сфери життєдіяльності, зокрема й у медицину. Зростаючий обсяг медичної інформації створює необхідність у ефективному управлінні медичними даними, які включають відомості про пацієнтів, історії хвороб, результати діагностичних процедур та іншу важливу інформацію. Медичні установи, приватні лікарські кабінети та окремі фахівці все частіше застосовують цифрові системи як зручний спосіб організації, зберігання, обробки та швидкого доступу до медичних записів.

Актуальність дослідження.

Потреба в розробці зручного, масштабованого та функціонального програмного забезпечення для управління медичними записами зумовлює актуальність цього дослідження. Більшість існуючих рішень або надто складні для впровадження та використання, або не враховують специфіку локального застосування у невеликих клініках та лікарських кабінетах. Саме тому створення десктопного клієнт-серверного застосунку, що підтримує гнучке адміністрування, електронні медичні картки пацієнтів, доступ до медичних записів та ефективний пошук інформації, має значну практичну цінність.

Мета розробки.

Ціль розробки програмного забезпечення – створити десктопну інформаційну систему, що забезпечує повний цикл роботи з електронними медичними даними: від створення та редагування пацієнтських карток до збереження результатів обстежень, аналізів і формування медичних звітів.
Рішення орієнтоване на широке коло користувачів, включаючи лікарів, медичних сестер та адміністративний персонал. У процесі розробки враховано

як потреби адміністраторів системи, які здійснюють управління базою даних, так і кінцевих користувачів, які переглядають, створюють і змінюють записи.

Використані технології.

Для досягнення поставленої мети використано сучасні технології, що гарантують гнучкість і надійність програмного забезпечення:

- C# для реалізації клієнтської частини у середовищі Windows Forms, що дозволяє створити зрозумілий інтерфейс користувача;
- MySQL [7] як система керування базами даних для зберігання структурованої медичної інформації;
- додаткові інструменти для перевірки коректності даних, обробки запитів, управління правами доступу, забезпечення безпеки автентифікації та авторизації користувачів.

Результати розробки.

Розроблена система пройшла тестування в умовах імітованого медичного середовища, отримавши позитивні відгуки щодо зручності використання та функціональності. За результатами дослідження підготовлено тези, які детально описують принципи побудови системи, її архітектурні рішення та ключові функціональні можливості.

Структура пояснювальної записки.

Пояснювальна записка дипломної роботи містить:

- 73 сторінки із детальним описом розробки.
- додатки, що включають фрагменти програмного коду, приклади SQL-запитів тощо.

Основні розділи роботи.

- системний аналіз предметної області – постановка задачі, аналіз аналогічних систем, моделювання предметної області;
- інформаційне забезпечення – опис логічної моделі даних, обґрунтування вибору СУБД, створення інформаційної бази;

- прикладне програмне забезпечення – структура програми, вибір інструментарію, реалізація модулів і функціоналу;
- рекомендації щодо впровадження та експлуатації – результати тестування, вимоги до середовища розгортання, інсталяційний пакет і рекомендації з використання.

Отже, представлена робота спрямована на вирішення важливої практичної задачі – створення ефективного, доступного та безпечного інструменту для управління електронними медичними записами, що відповідає потребам сучасних медичних установ і фахівців у сфері охорони здоров’я.

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

У сучасних умовах інтенсивного розвитку цифрових технологій та активного переходу багатьох сфер діяльності в онлайн-середовище, особливої значущості набуває автоматизація медичних процесів, зокрема створення та впровадження інформаційних систем для управління електронними медичними даними. Інформаційна система управління медичними записами — це спеціалізована програмна платформа, яка забезпечує зберігання, облік, опрацювання та доступ до медичної інформації про пацієнтів, їхні історії хвороб, результати обстежень, діагнози, призначення та інші важливі медичні відомості.

Предметна область включає функціонування електронної медичної системи в умовах закладів охорони здоров’я — як державних, так і приватних, — з можливістю обліку пацієнтів, ведення електронної медичної документації, управління правами доступу, фіксації візитів, обліку процедур, контролю за призначеннями та формування звітної документації. З огляду на зростаючі вимоги до точності, швидкості опрацювання та безпеки персональних даних, подібні системи стають не просто зручним інструментом, а необхідною складовою сучасної медичної інфраструктури.

У межах предметної області ключовими об’єктами є:

- користувачі (лікарі, адміністратори);
- пацієнти (з унікальними медичними картками);
- медичні записи (відвідування, симптоми, діагнози, результати аналізів, призначення);
- процеси реєстрації, авторизації та управління доступом;
- функції пошуку та фільтрації пацієнтів чи записів (за ПІБ, діагнозом, датою);
- модулі введення, редагування, перегляду та видалення записів;

- механізми формування звітів, виписок, направлень;
- системні журнали активності користувачів і реєстрації дій.

Медичні інформаційні системи можуть бути як частиною великої лікарняної структури, так і рішенням для приватної лікарської практики. Залежно від масштабу, система здатна обробляти тисячі медичних карток, взаємодіяти з іншими інформаційними сервісами , підтримувати інтеграцію з хмарними сервісами, системами резервного копіювання, а також відповідати стандартам безпеки персональних даних .

У рамках даної предметної області важливим є побудова ефективної архітектури інформаційної системи, що забезпечує швидкий доступ до даних, захищене зберігання медичної інформації, можливість масштабування, простоту адміністрування та стабільну роботу в локальній або мережевій інфраструктурі. Особливо актуальними завданнями є: реалізація багаторівневої автентифікації, ведення історії змін у записах, формування автоматизованих звітів, підтримка пошуку за медичними критеріями, резервне копіювання та захист від несанкціонованого доступу.

Таким чином, предметна область охоплює сукупність процесів і компонентів, спрямованих на побудову, підтримку та розвиток електронної системи ведення медичних записів, що є підґрунтям для подальшого аналізу, моделювання та розробки програмного забезпечення для автоматизованого управління медичною інформацією.

1.2 Аналіз вимог до програмної системи

Аналіз вимог є визначальним етапом системного аналізу, що дає змогу сформувати чітке розуміння функціональних та нефункціональних характеристик майбутньої інформаційної системи, ідентифікувати категорії користувачів, сценарії їхньої взаємодії з системою та очікувані результати від її

впровадження. Саме на основі цього етапу здійснюється проєктування та подальша розробка програмного забезпечення.

Загальна характеристика системи

Інформаційна система для управління медичними записами належить до класу прикладних автоматизованих інформаційних систем, призначених для обліку, опрацювання, зберігання та надання доступу до медичної інформації. Система має на меті автоматизувати рутинні адміністративно-лікарняні процеси, полегшити роботу медичного персоналу, забезпечити постійний і швидкий доступ до історій хвороб, результатів обстежень, діагнозів та інших важливих даних.

За характером оброблюваної інформації розроблена система є орієнтованою на електронні картки пацієнтів, а за масштабом — застосовною як для індивідуального використання в кабінеті приватного лікаря, так і для мережевої взаємодії в межах медичного закладу. Система передбачає повний цикл роботи з медичною документацією — від реєстрації пацієнта до проведення прийомів, формування звітності та архівування інформації.

Функціональні вимоги

Відповідно до визначеної мети, система повинна реалізовувати наступні основні функції:

- реєстрація та ідентифікація користувачів (лікарів, медичних сестер, адміністраторів);
- розмежування прав доступу до функціональних можливостей;
- створення та редагування електронних медичних карток пацієнтів;
- ведення історії захворювань: симптоми, діагнози, результати аналізів, призначення лікування;
- запис і планування візитів;
- пошук пацієнтів за ПІБ, діагнозом, датою візиту тощо;

- перегляд повної історії відвідувань та призначень;
- видалення та архівація записів (з правами адміністратора);
- формування звітів: кількість візитів, частота захворювань, активність користувачів;
- ведення журналу подій для відстеження змін і переглядів;
- адміністрування користувачів і системних налаштувань.

Нефункціональні вимоги

Окрім реалізації основного функціоналу, система повинна відповідати таким нефункціональним вимогам:

- зручність використання (юзабіліті): інтуїтивно зрозумілий інтерфейс, оптимізований для швидкої навігації та щоденної роботи персоналу;
- продуктивність: швидке опрацювання запитів, доступ до карток, пошук інформації;
- надійність: збереження даних навіть у випадку аварійних ситуацій, перевірка коректності введених даних;
- безпека: захист медичної та персональної інформації, обмеження доступу, шифрування даних;
- масштабованість: можливість розширення системи зі збільшенням кількості пацієнтів та користувачів;
- підтримка та супроводжуваність: зручність оновлення та вдосконалення, підтримка додаткових модулів.

Таким чином, сформульовані вимоги визначають загальну архітектуру, функціональні межі, технічні характеристики та експлуатаційні параметри майбутньої системи управління медичними записами. Ці вимоги є основою для побудови структурної моделі, розробки функціональних модулів та створення повноцінного, надійного програмного продукту, здатного забезпечити ефективну роботу медичного персоналу.

1.3 Моделювання предметної області

Для забезпечення повного і чіткого розуміння логіки функціонування інформаційної системи управління медичними записами доцільно використовувати засоби об’єктно-орієнтованого моделювання предметної області. Основним інструментом такого моделювання є мова UML (Unified Modeling Language), яка забезпечує стандартизований спосіб візуалізації проектування систем. UML дозволяє описати функціональність, структуру та поведінку системи за допомогою графічних діаграм.

Діаграма прецедентів (Use Case Diagram)

Діаграма прецедентів використовується для [1] візуалізації функціональних вимог до системи через сценарії взаємодії користувачів (акторів) із системою. На рис.1 показано взаємодію двох основних акторів: адміністратора та лікаря з функціональністю медичної системи.

Адміністратор має доступ до таких функцій:

- реєстрація пацієнтів у системі;
- редагування особистих даних пацієнтів;
- перегляд історії пацієнтів;

Лікар має доступ до наступних функціональних можливостей:

- перегляд історії пацієнтів;
- створення медичних записів;
- додавання діагнозів;
- призначення рецептів.

Усі ці дії взаємодіють із базою даних [7] *MySQL: medical_clinic*, яка забезпечує зберігання та обробку інформації про пацієнтів, їхній стан здоров’я та медичну історію.

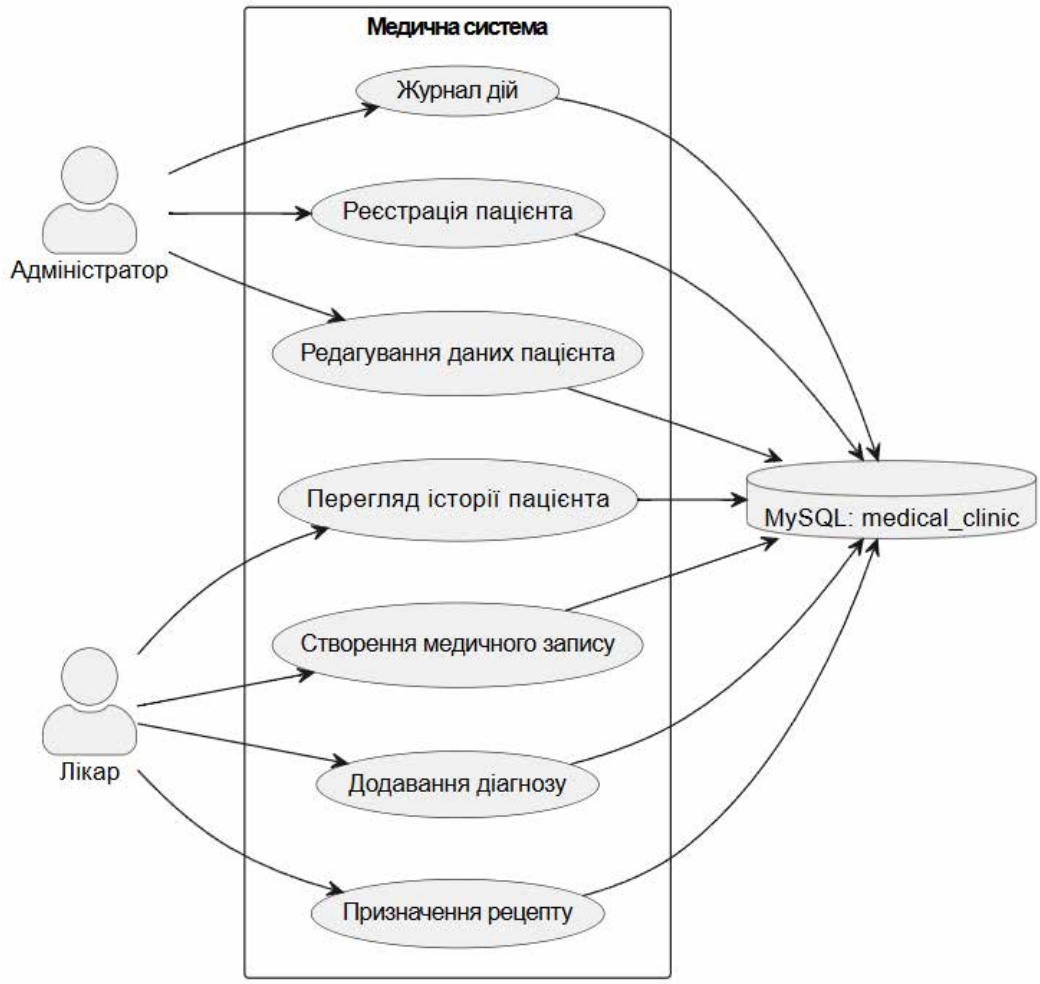


Рис.1 Діаграма прецедентів користувачів системи

Діаграма діяльності (Activity Diagram)

Для візуалізації логіки виконання ключових сценаріїв взаємодії користувачів із системою управління медичними записами доцільно застосовувати діаграми діяльності [2] UML. На рисунку 2 представлено діаграму діяльності одного з основних сценаріїв використання системи — «Створення медичного запису для пацієнта».

Сценарій починається з того, що адміністратор, після проходження авторизації, здійснює реєстрацію нового пацієнта. Дані нового користувача заносяться до бази даних, а сам факт реєстрації фіксується у журналі подій. Далі

до процесу приєднується лікар, який авторизується в системі, обирає пацієнта та ініціює створення нового медичного запису.

У межах цього запису лікар вносить супровідні нотатки, встановлює діагноз — обираючи наявний із бази або створюючи новий, а також призначає лікування у вигляді рецепту. Усі ці дії зберігаються у відповідних таблицях бази даних (medicalrecord, diagnosis, prescription) і пов’язуються між собою через унікальні ідентифікатори. [12]

Після завершення формування запису система зберігає внесену інформацію та відображає повідомлення про успішне завершення операції. Усі ключові дії, виконані в процесі, реєструються для подальшого аудиту та забезпечення безпеки даних.

Уся логіка — від реєстрації пацієнта до збереження медичного запису — чітко представлена у вигляді діаграми діяльності на рис. 2, яка демонструє послідовність дій, логіку перевірок і взаємодію між користувачами та інформаційною системою.

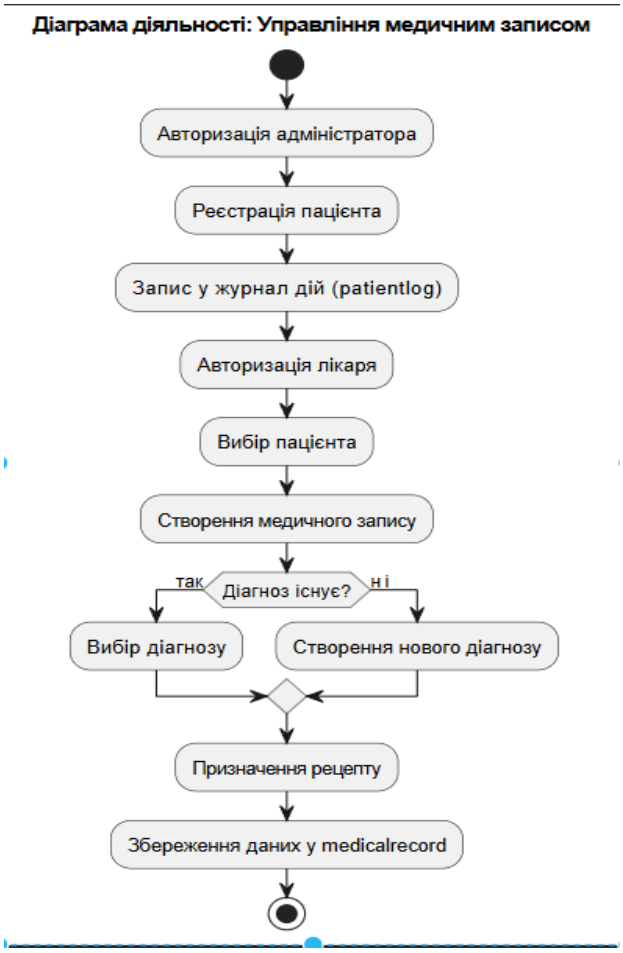


Рис.2 Діаграма діяльності

Діаграма послідовності (Sequence Diagram)

Для наочного відображення взаємодії між об’єктами системи в межах одного з ключових сценаріїв використовується [3] **діаграма послідовності (додаток Б)**. На додатку Б представлено процес створення нового медичного запису пацієнта з боку адміністратора та лікаря.

Сценарій починається з того, що адміністратор відкриває інтерфейс реєстрації пацієнта. Після цього вводяться основні дані пацієнта, які надсилаються до серверної частини системи. Сервер обробляє запит, створює запис у таблиці пацієнтів (patient) і фіксує відповідну дію в журналі подій (patientlog). Після успішного завершення реєстрації адміністратор отримує підтвердження.

Далі до процесу долучається лікар. Він авторизується, обирає пацієнта зі списку та відкриває форму створення медичного запису. У межах цього етапу лікар заповнює дані про візит, зазначає діагноз (вибираючи з бази або створюючи новий), а також за потреби додає рецепт. Введена інформація надсилається до сервера, який перевіряє її на відповідність вимогам.

У разі виявлення помилки система надсилає повідомлення з описом проблеми. Якщо ж усі дані коректні, система зберігає медичний запис до таблиці medicalrecord, а також пов’язані з ним записи у таблицях diagnosis і prescription. Завершальним етапом є повідомлення лікаря про успішне створення запису.

Використання UML-діаграм [1] дає змогу формалізувати логіку функціонування системи, чітко окреслити ролі користувачів, їхні дії та структуру даних. Це суттєво полегшує етапи розробки, тестування та подальшого обслуговування програмного забезпечення.

1.4 Огляд інформаційних джерел та існуючих рішень

У процесі створення інформаційної системи управління медичними записами доцільним є проведення аналізу існуючих програмних рішень, які виконують подібні функції в сфері електронного документообігу в медичних установах. Такий аналіз дає змогу не лише ознайомитися з актуальними підходами до реалізації систем електронної охорони здоров’я, але й виявити сильні та слабкі сторони вже наявних рішень, що, в свою чергу, дозволяє уникнути поширених помилок та запозичити успішні архітектурні та інтерфейсні рішення.

Порівняння подібних систем допомагає:

- з’ясувати, які саме функції є найбільш затребуваними серед цільових користувачів (лікарів, медсестер, адміністративного персоналу);

- виявити типові модулі (наприклад, облік пацієнтів, ведення історії хвороб, звітність, планування прийомів), які мають бути реалізовані у власному продукті;
- оцінити зручність інтерфейсу, рівень автоматизації процесів, можливості кастомізації та інтеграції з іншими системами (наприклад, лабораторними модулями або електронними сервісами охорони здоров’я);
- проаналізувати рівень захисту персональних даних, реалізацію системи доступу до конфіденційної інформації, підтримку нормативних вимог.

У результаті такого дослідження стає можливим сформувати обґрунтовані вимоги до майбутньої системи, оптимально визначити її функціональну структуру, ключові сценарії використання та основні етапи розробки.

Нижче подано аналіз чотирьох програмних продуктів, які частково або повністю реалізують функціональність, пов’язану з веденням електронних медичних записів, обліком пацієнтів і документацією, що є важливим джерелом натхнення та практичних орієнтирів для створення ефективного і сучасного рішення.

Medesk

Medesk (рис.4) — хмарна медична інформаційна платформа, яка активно використовується в приватних клініках, діагностичних центрах та медичних кабінетах. Вона забезпечує централізоване зберігання медичної інформації, включаючи електронні картки пацієнтів, планування прийомів, ведення історії хвороб, результати аналізів, а також надає зручний доступ до фінансової звітності та статистики. Система має зручний інтерфейс, підтримує інтеграцію з лабораторними модулями та CRM-системами. Хмарна інфраструктура дозволяє працювати з будь-якого пристрою, а функціонал повідомлень та нагадувань робить комунікацію з пацієнтами ефективною та сучасною.

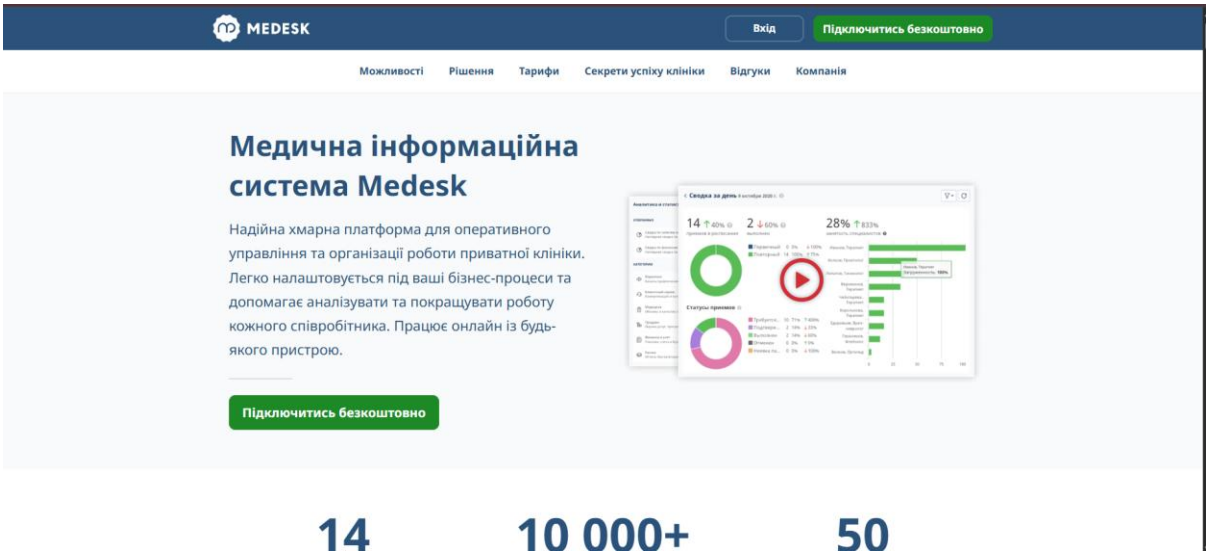


Рис.4.1 Головна сторінка Medesk

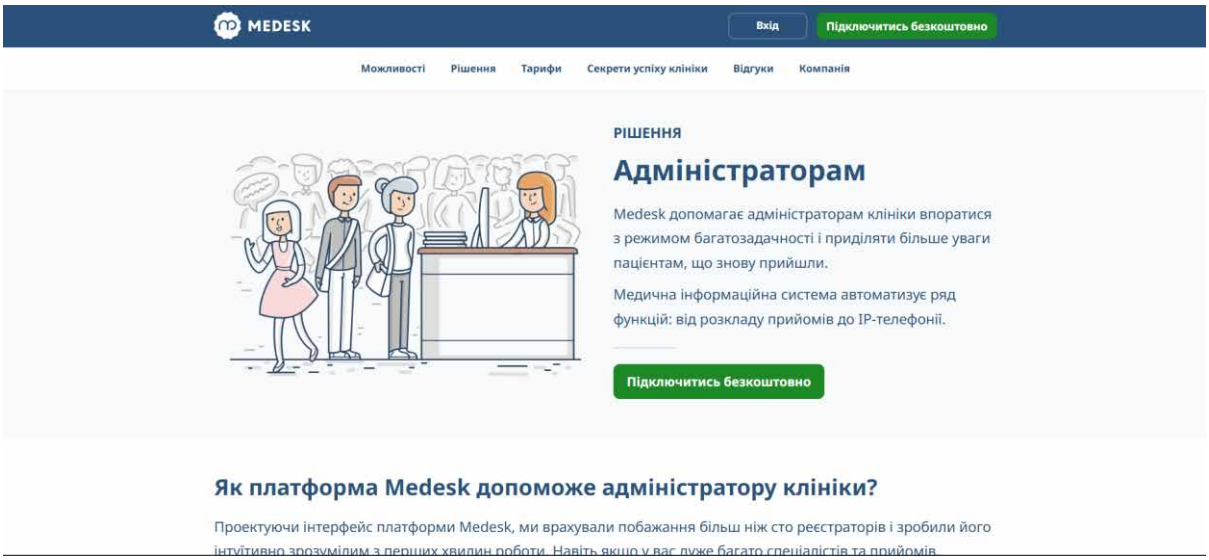


Рис.4.2 Вкладка рішення в Medesk

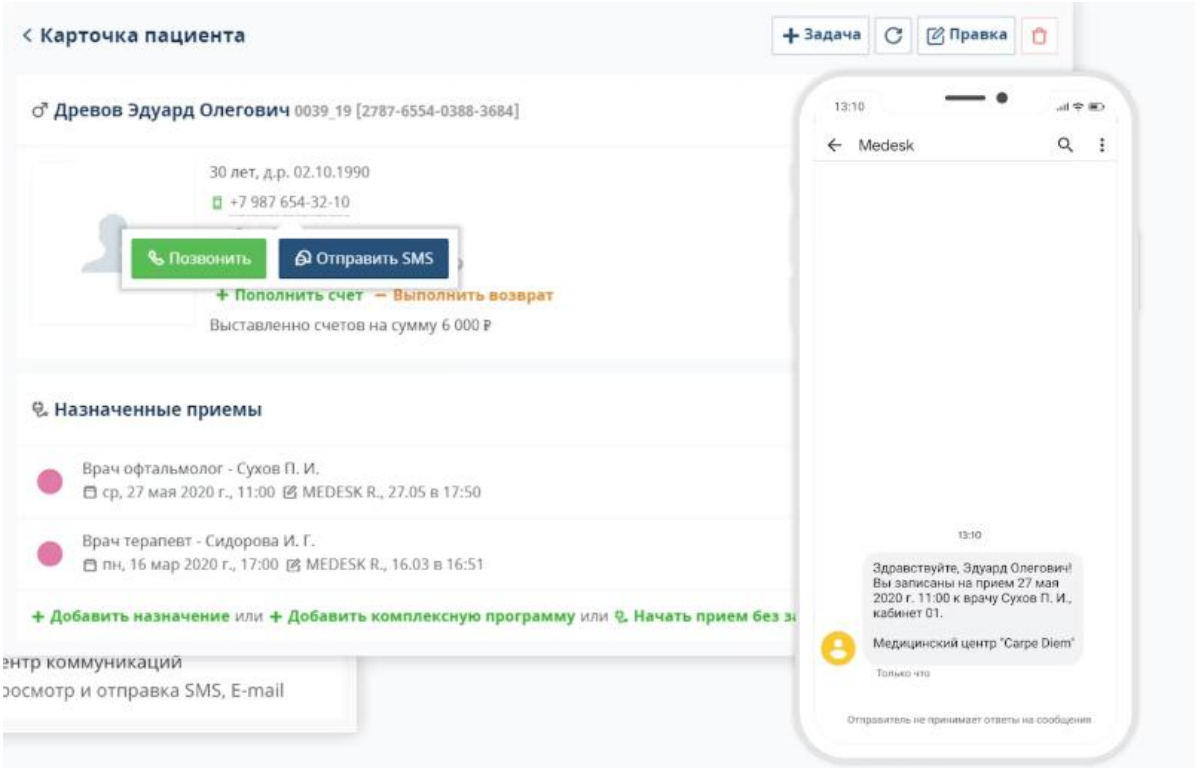


Рис.4.3 Картка пацієнта в Medesk

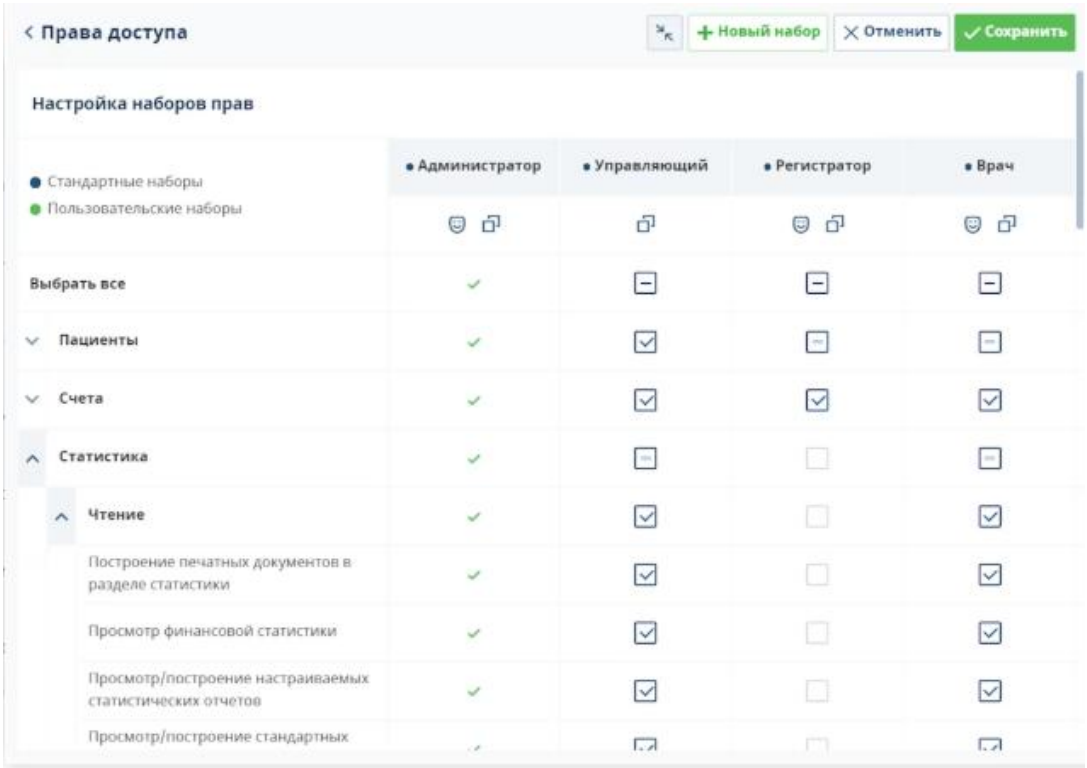


Рис.4.4 Вікно налаштування прав доступу в Medesk

SimplexMed

SimplexMed (рис.5) — українська система, що орієнтована на автоматизацію обліку пацієнтів, прийомів і медичної документації у приватній медичній практиці та невеликих клініках. Вона дозволяє створювати і зберігати електронні картки пацієнтів, вести історію хвороб, друкувати рецепти, довідки та направлення, а також організовувати графік роботи лікарів. Присутні модулі для адміністративного обліку, аналітики, фінансового контролю. Система відповідає українським нормативним вимогам, має інтерфейс українською мовою та підтримує гнучке налаштування прав доступу для різних категорій медичного персоналу.

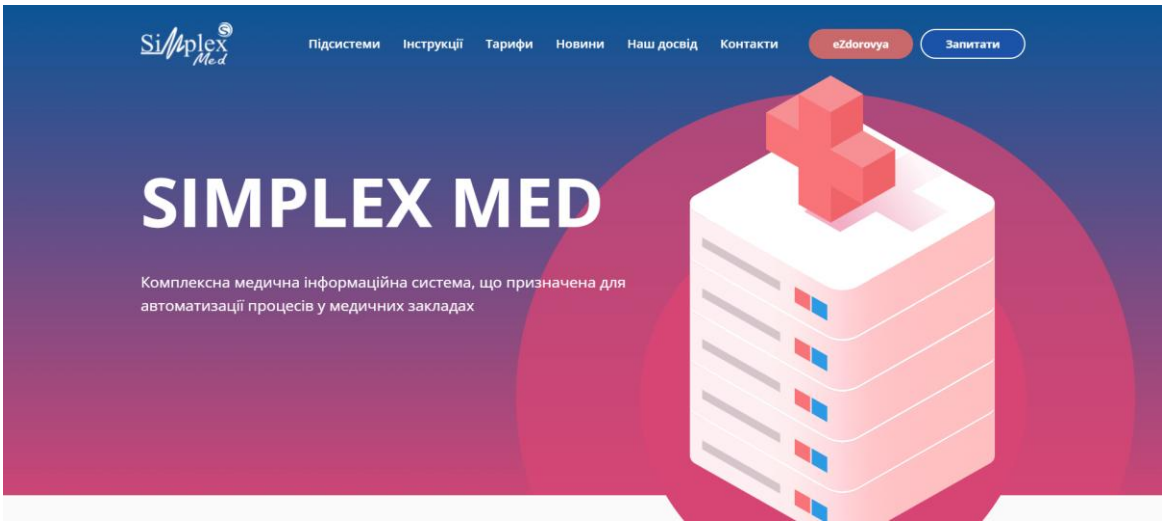


Рис.5.1 Головна сторінка SimplexMed

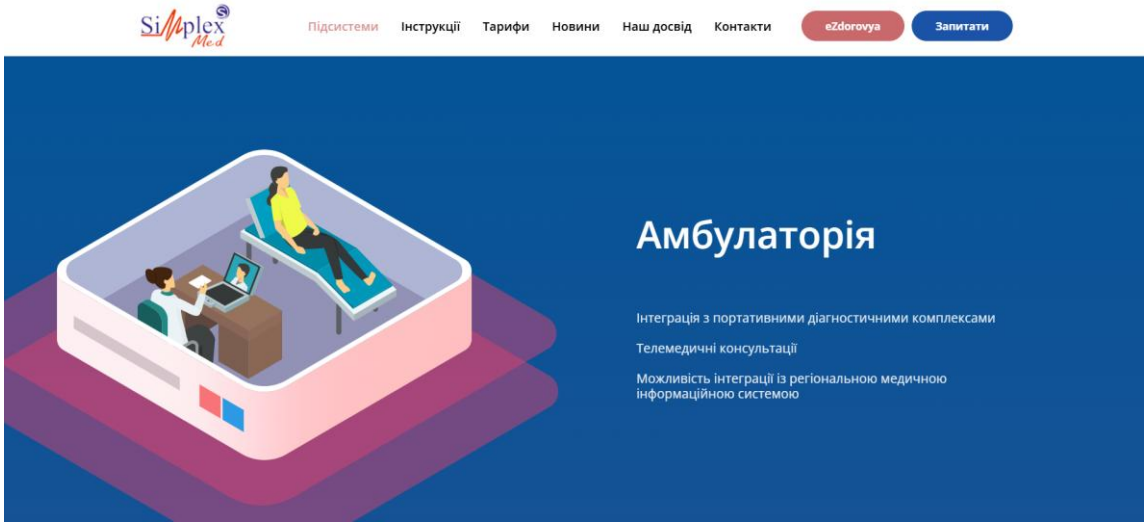


Рис.5.2 Інтерфейс підсистеми SimplexMed

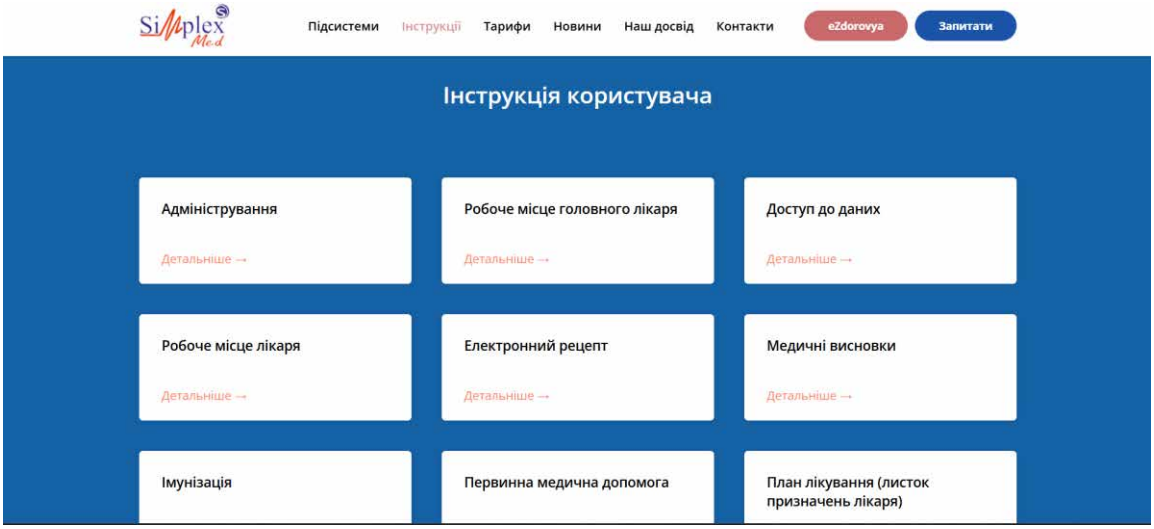


Рис.5.3 Інтерфейс інструкцій користувача SimplexMed

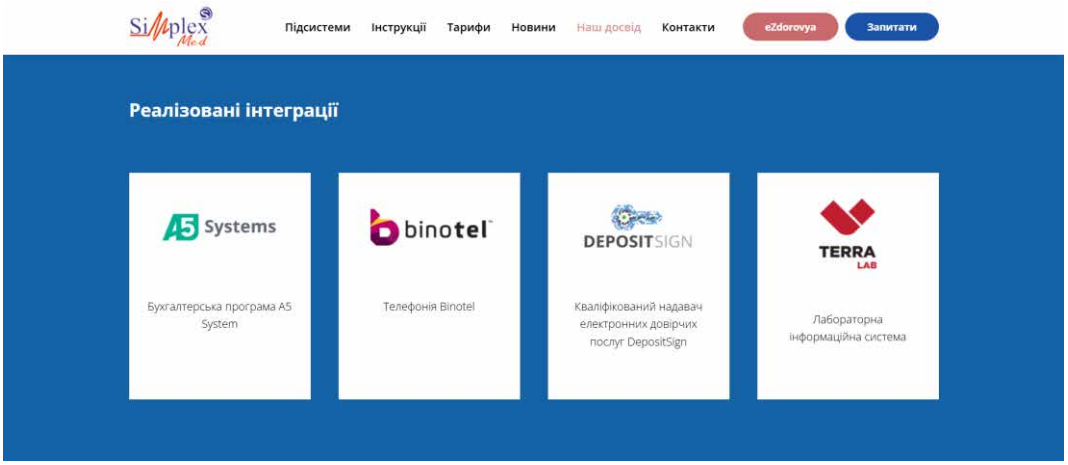


Рис.5.4 Інтерфейс інтеграцій SimplexMed

OpenEMR

OpenEMR (рис.6) — одну з найпоширеніших безкоштовних систем управління медичною інформацією, яка використовується по всьому світу. OpenEMR підтримує повний цикл медичного документообігу, включаючи ведення електронних медичних записів, управління розкладом прийомів, обробку страхових випадків, формування звітів, а також взаємодію з лабораторними службами. Завдяки відкритому коду, систему можна адаптувати під будь-які потреби конкретного закладу. Вона відповідає міжнародним

стандартам, включаючи HIPAA, і має багатомовну підтримку, що робить її зручною для впровадження в різних країнах.

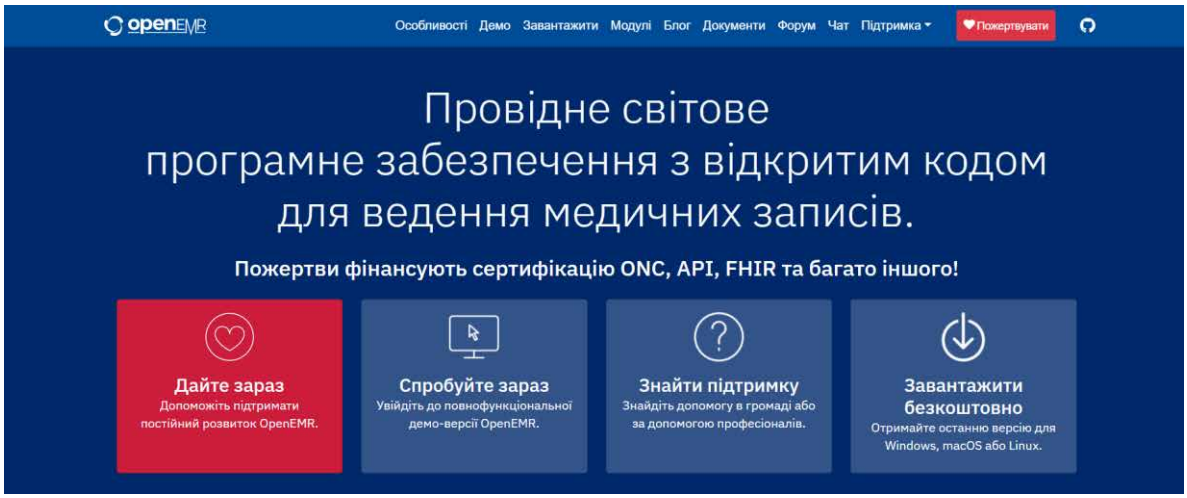


Рис.6.1 Головна сторінка OpenEMR

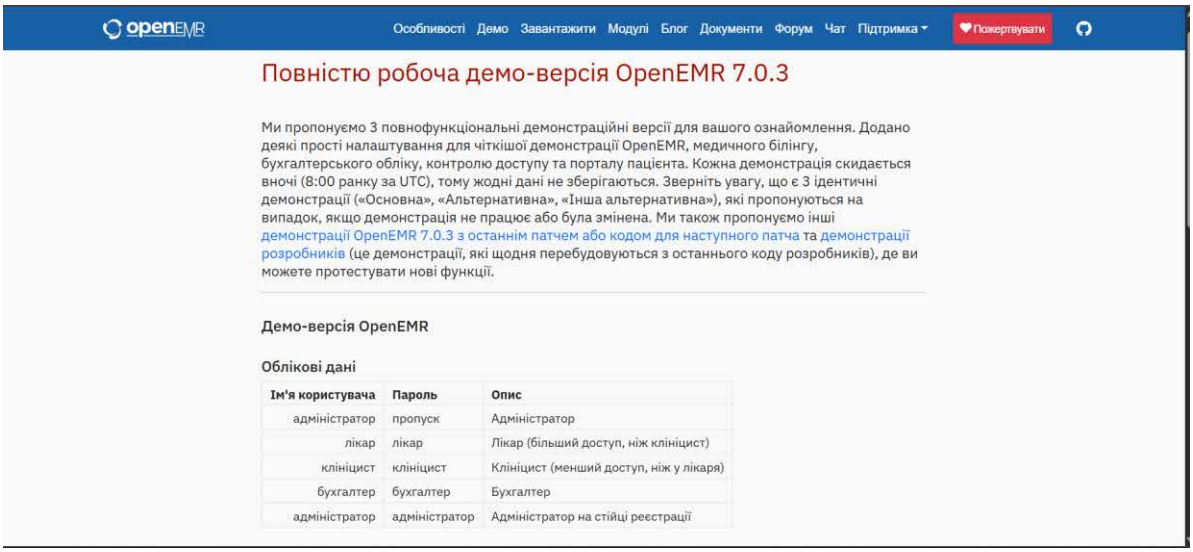


Рис. 6.2 Інтерфейс сторінки демо-версії OpenEMR

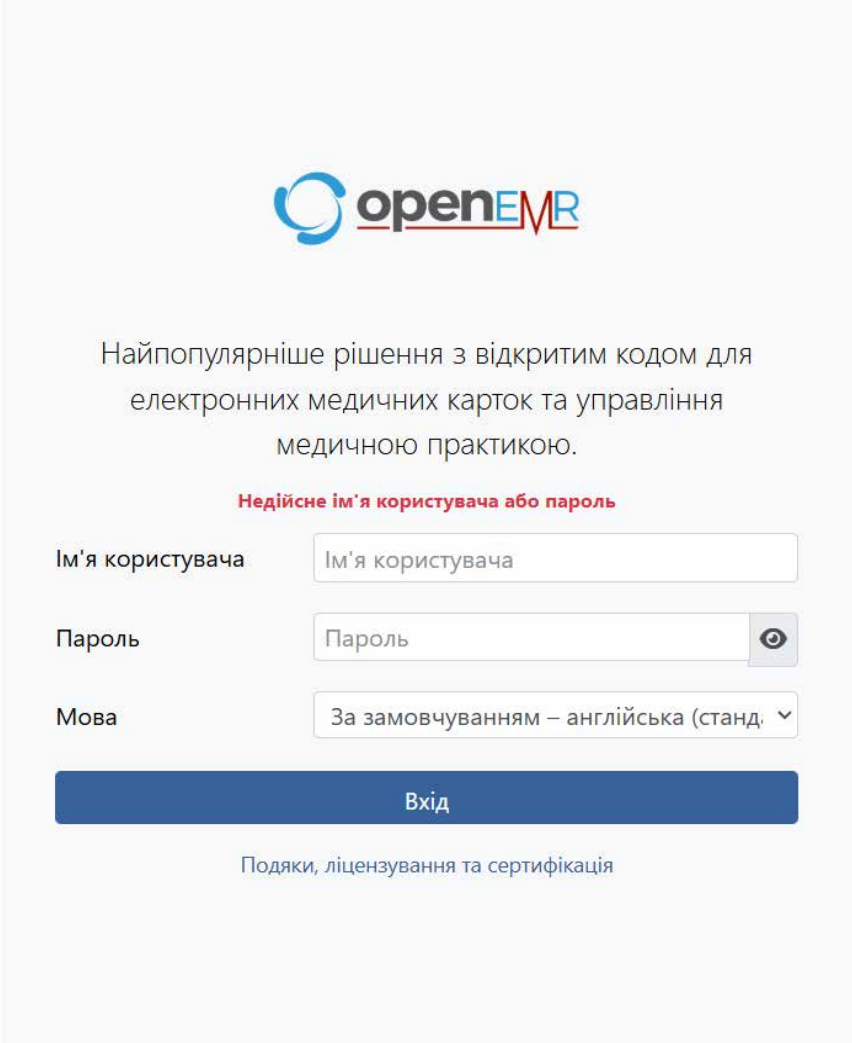


Рис. 6.3 Вікно авторизації користувача в OpenEMR

Рис. 6.4 Сторінка завантаження стабільної версії OpenEMR

Health24

Health24 (рис.7)— це сучасна хмарна медична інформаційна система, розроблена для автоматизації повного циклу управління медичними закладами різних форм власності, включаючи приватні клініки, державні установи та лікарів-ФОП. Система надає можливість створення персонального веб-сайту, онлайн-запису пацієнтів, управління візитами, доступу до всіх можливостей eHealth, а також мобільного додатку для пацієнтів.

Health24 інтегрується з EMCIlab, що покращує роботу з аналізами та лабораторними дослідженнями. Для державних установ первинної, вторинної та третинної спеціалізованої медичної допомоги система пропонує всі модулі eHealth для роботи відповідно до нових вимог НСЗУ.

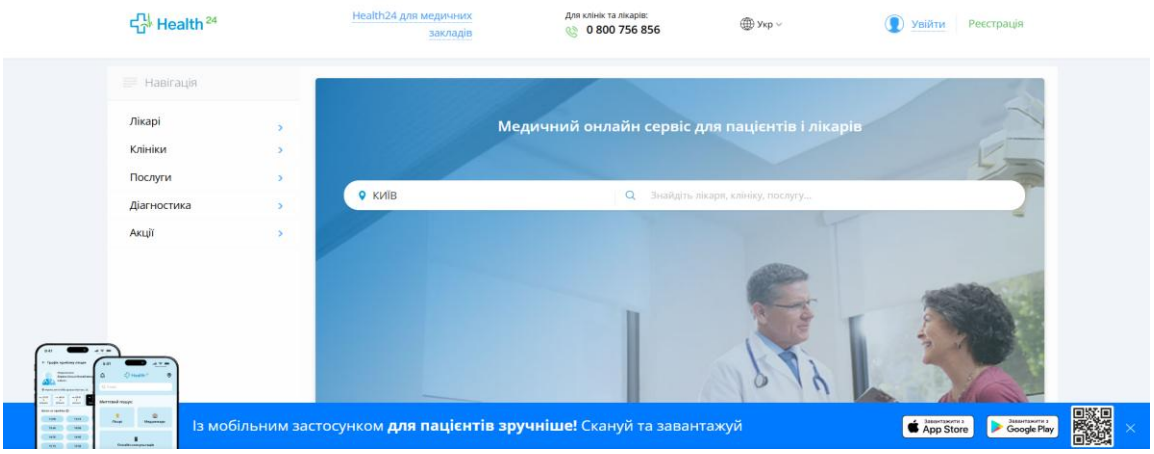


Рис.7.1 Головна сторінка Health24

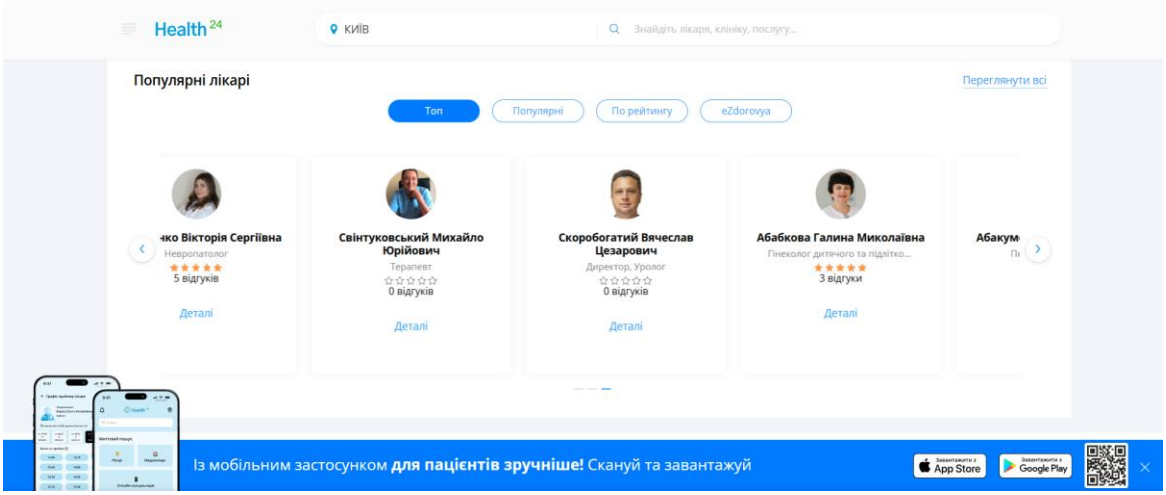


Рис.7.2 Інтерфейс розділу з переліком популярних лікарів у Health24

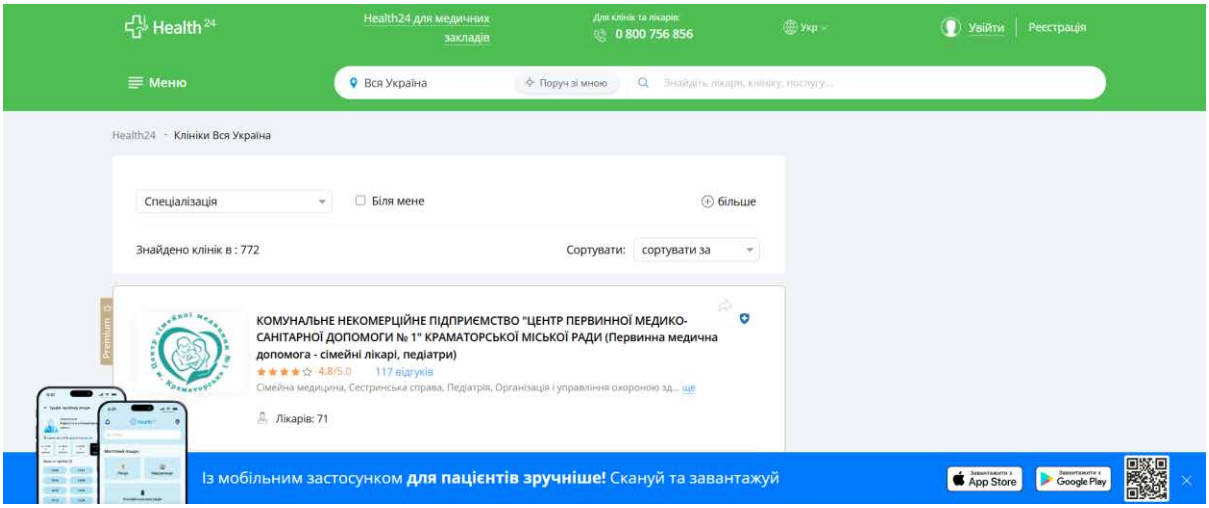


Рис.7.3 Інтерфейс пошуку лікарів у Health24

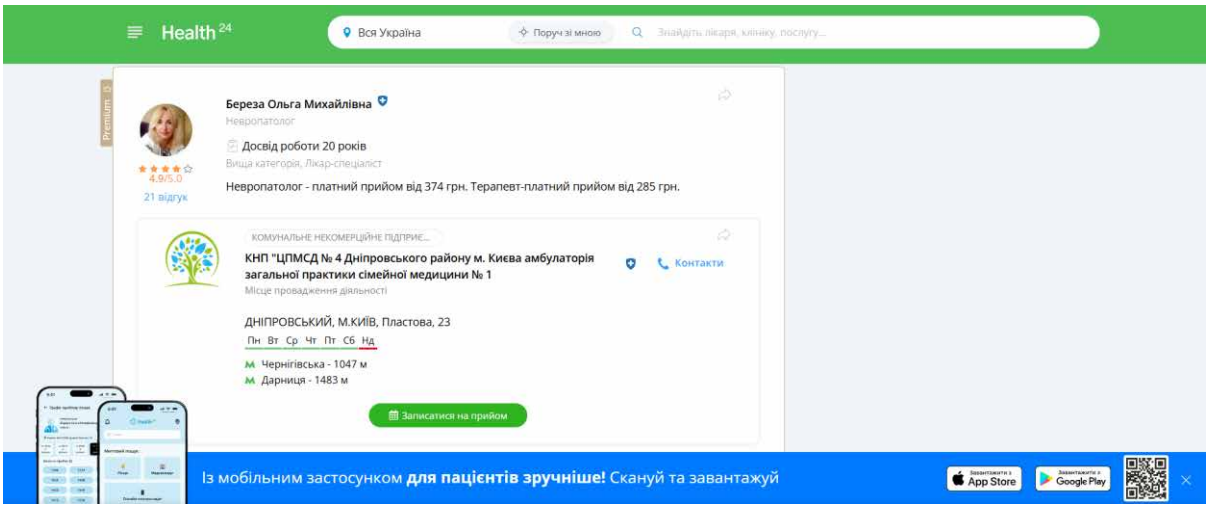


Рис.7.4 Сторінка профілю лікаря у системі Health24

Таблиця 1

Порівняльна таблиця програм аналогів

Критерій	Health24	Medesk	SimplexMed	OpenEMR
Тип	Хмарна система	Хмарна/десктопна	Хмарна система	Десктопна/хмарна
Доступ до записів	Онлайн	Онлайн	Онлайн	Онлайн/локально
Модулі (функціональність)	Записи, аналітика, звіти	Записи, телемедицина, фінанси	Записи, управління пацієнтами	Повна EHR-функціональність
Пошук/фільтрація	Так	Так	Так	Так
Підтримка електронної карти	Так	Так	Так	Так
Управління профілем пацієнта	Так	Так	Так	Так
Медичні розклади, візити	Так	Так	Так	Так
Платформа	Веб	Веб, Windows	Веб	Windows, Linux, Web
Ліцензія	Платна	Платна	Платна	Безкоштовна (open-source)
Підтримка української мови	Частково/ні	Частково/так	Так	Так (може потребувати налаштування)
Інтеграція з лабораторіями	Обмежена	Так	Так	Так
Синхронізація з хмарою	Так	Так	Так	Так

1.5 Постановка завдання

Метою даної дипломної роботи є проєктування та розробка інформаційної системи Medix — десктопного застосунку для управління електронними медичними записами, який забезпечує ефективну роботу з пацієнтськими

картками, облік медичних візитів, фіксацію діагнозів, призначень, результатів аналізів, а також адміністрування бази даних пацієнтів і користувачів медичного закладу.

Система призначена для локального використання в межах невеликих медичних установ, приватних кабінетів або амбулаторій. Передбачається, що застосунок забезпечить автоматизацію основних медико-адміністративних процесів, підвищить точність та швидкість обробки інформації, а також забезпечить зручний доступ до структурованих медичних даних.

Інформаційна система [3] повинна реалізовувати наступні функціональні можливості:

Для лікаря:

- реєстрація нового пацієнта;
- перегляд і редагування даних пацієнта;
- формування та збереження електронної медичної картки;
- додавання записів про візити, діагнози, симптоми, результати аналізів;
- призначення лікування та формування направлень
- пошук пацієнтів за ПІБ, номером картки або діагнозом.;
- перегляд історії хвороби пацієнта.

Для адміністратора:

- керування обліковими записами користувачів системи;
- контроль змін у базі даних.

Нефункціональні вимоги:

- архітектура: клієнт-серверна модель із підключенням до бази MySQL;
- платформа: Windows-додаток, реалізований із використанням .NET (WinForms);
- інтерфейс: багатовкладковий, локалізований (українська мова), з підтримкою швидкої навігації, пошуку та повідомлень про помилки;
- збереження даних: централізоване, у реляційній базі з підтримкою зв’язків між таблицями (пацієнти, візити, діагнози, результати тощо);

- безпека: валідація введених даних, базова автентифікація користувачів, захист доступу за ролями, збереження конфіденційності медичної інформації;
- масштабованість: модульна структура, що дозволяє розширення функціональності — додавання лабораторного модуля, телемедицини, синхронізації з хмарними сервісами.

Розробка вважається завершеною, якщо:

- реалізовано всі заявлені функціональні модулі;
- програма стабільно працює з базою MySQL;
- пройдено модульне й інтеграційне тестування;
- підготовлено базову документацію користувача;
- забезпечено успішну установку програми на ОС Windows 10 і вище.

Таким чином, результатом виконання цієї роботи стане прототип інформаційної системи Medix, який дозволить ефективно вести електронну документацію в медичних закладах, оптимізуючи рутинні процеси та створюючи надійну платформу для подальшого розвитку функціоналу.

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Логічна модель даних

Логічна модель даних Medix є основою структури інформаційної бази системи управління медичними записами. Вона забезпечує ефективне зберігання, обробку та доступ до медичної інформації пацієнтів. Модель побудована відповідно до принципів реляційної моделі та нормалізована до третьої нормальної форми (3NF), що виключає надлишковість та підвищує логічну цілісність даних.

ER-модель (Entity-Relationship) [1], зображена на додатку В, використана при побудові, відображає основні сутності предметної області охорони здоров’я, їх атрибути та зв’язки. Такий підхід дозволяє проектувати універсальну структуру бази даних, незалежну від конкретної СУБД.

Основні сутності системи Medix та їх атрибути:

- **Patient** — зберігає персональні дані пацієнтів:
  PatientID (PK), LastName, FirstName, MiddleName, MedicalCardNumber, Address, Email, Password.
- **Doctor** — інформація про лікарів:
  DoctorID (PK), LastName, FirstName, MiddleName, Position, PhoneNumber, Email, Password, SpecializationID (FK).
- **Administrator** — адміністратори медичного закладу:
  AdminID (PK), LastName, FirstName, MiddleName, Position, PhoneNumber, Email, Password.
- **Diagnosis** — довідник діагнозів:
  DiagnosisID (PK), Name, Description.
- **MedicalRecord** — медичні записи пацієнтів:
  RecordID (PK), OpeningDate, Notes, DiagnosisConclusion, PatientID (FK), DiagnosisID (FK).

- **Prescription** — призначення лікарських засобів:
  PrescriptionID (PK), RecordID (FK), PrescriptionDate, Medication, Dosage, Instructions.

- **Appointment** — облік прийомів пацієнтів у лікарів:
  AppointmentID (PK), PatientID (FK), DoctorID (FK), AdminID (FK), AppointmentDate, Purpose, Status.

- **PatientLog** — журнал дій пацієнтів у системі:
  LogID (PK), PatientID (FK), ActionType, ActionTime.

- **Specialization** — класифікатор спеціалізацій лікарів:
  SpecializationID (PK), Name.

2.2 Вибір системи управління інформаційною базою

На цьому етапі проєктування системи було здійснено обґрунтований вибір системи управління базами даних (СУБД), яка є ключовим компонентом для зберігання, обробки та впорядкування медичної інформації. Для реалізації інформаційної бази програмного забезпечення було обрано MySQL — надійну та продуктивну реляційну СУБД з відкритим вихідним кодом, яка широко застосовується в медичних, комерційних та державних системах.

MySQL [7] є однією з найпопулярніших СУБД у світі, і її вибір обумовлений низкою вагомих переваг. Зокрема, вона забезпечує високу швидкодію при виконанні запитів, підтримує складні SQL-оператори, транзакції та індекси, що дає змогу ефективно працювати з великими обсягами структурованих даних. MySQL має хорошу підтримку механізмів цілісності даних, включаючи зовнішні ключі, каскадні операції та обмеження, що є особливо важливим при зберіганні взаємопов’язаних медичних записів (наприклад, візити, діагнози, аналізи, пацієнти).

Крім цього, MySQL легко інтегрується з середовищем розробки .NET, що дозволяє швидко реалізувати зв’язок між додатком, написаним на C#

(WinForms), та базою даних. Гнучка модель прав доступу в MySQL дозволяє обмежити доступ до певних даних залежно від ролі користувача — лікаря, медсестри або адміністратора. У поєднанні з механізмами автентифікації на рівні застосунку це забезпечує високий рівень безпеки конфіденційної інформації.

Серед інших переваг MySQL можна відзначити:

- кросплатформенність і легкість у розгортанні;
- наявність широкого спектра інструментів адміністрування (MySQL Workbench, phpMyAdmin тощо);
- активну спільноту підтримки та наявність великої кількості документації;
- можливість масштабування при зростанні обсягу медичних даних.

Обрана технологія повністю відповідає поточним потребам системи Medix, а також відкриває перспективи для її подальшого розвитку — впровадження аналітичних модулів, резервного копіювання, інтеграції з лабораторними системами та іншими електронними медичними сервісами.

Таким чином, використання MySQL як основної СУБД для системи Medix дозволяє реалізувати стабільну, масштабовану та безпечну інформаційну платформу для зберігання та обробки електронних медичних записів, що відповідає як сучасним вимогам, так і майбутнім потребам у сфері охорони здоров’я.

2.3 Створення інформаційної бази

На основі розробленої концептуальної моделі інформаційної системи Medix було реалізовано фізичну структуру інформаційної бази із використанням СУБД MySQL (рис.9). Створення бази даних включало побудову основних та допоміжних таблиць, визначення зв’язків між ними, встановлення обмежень

цілісності (PRIMARY KEY, FOREIGN KEY), а також конфігурування типів даних та індексів відповідно до логіки предметної області.

У системі реалізовано такі ключові таблиці:

- **patient** — зберігає персональні дані пацієнтів:
  PatientID (PK), LastName, FirstName, MiddleName, MedicalCardNumber, Address, Email, Password.

- **doctor** — містить дані про лікарів:
  DoctorID (PK), LastName, FirstName, MiddleName, Position, PhoneNumber, Email, Password, SpecializationID (FK).

- **administrator** — інформація про адміністративний персонал:
  AdminID (PK), LastName, FirstName, MiddleName, Position, PhoneNumber, Email, Password.

- **diagnosis** — довідник діагнозів:
  DiagnosisID (PK), Name, Description.

- **medicalrecord** — медичні записи пацієнтів:
  RecordID (PK), OpeningDate, Notes, DiagnosisConclusion, PatientID (FK), DiagnosisID (FK).

- **prescription** — інформація про призначення:
  PrescriptionID (PK), RecordID (FK), PrescriptionDate, Medication, Dosage, Instructions.

- **appointment** — облік записів на прийом:
  AppointmentID (PK), PatientID (FK), DoctorID (FK), AdminID (FK), AppointmentDate, Purpose, Status.

- **patientlog** — журнал активності пацієнтів:
  LogID (PK), PatientID (FK), ActionType, ActionTime.

- **specialization** — довідник спеціалізацій лікарів:
  SpecializationID (PK), Name.

Реалізація у MySQL дозволила:

- здійснити жорстку перевірку цілісності даних за допомогою зовнішніх ключів;
- забезпечити масштабованість структури бази для подальшого розвитку;
- оптимізувати запити завдяки створенню індексів;
- підключати візуальні засоби адміністрування, такі як MySQL Workbench або phpMyAdmin, для побудови ER-діаграм і керування базою даних.

Повний SQL-скрипт створення таблиць із відповідними обмеженнями наведено у Додатку А до документації.

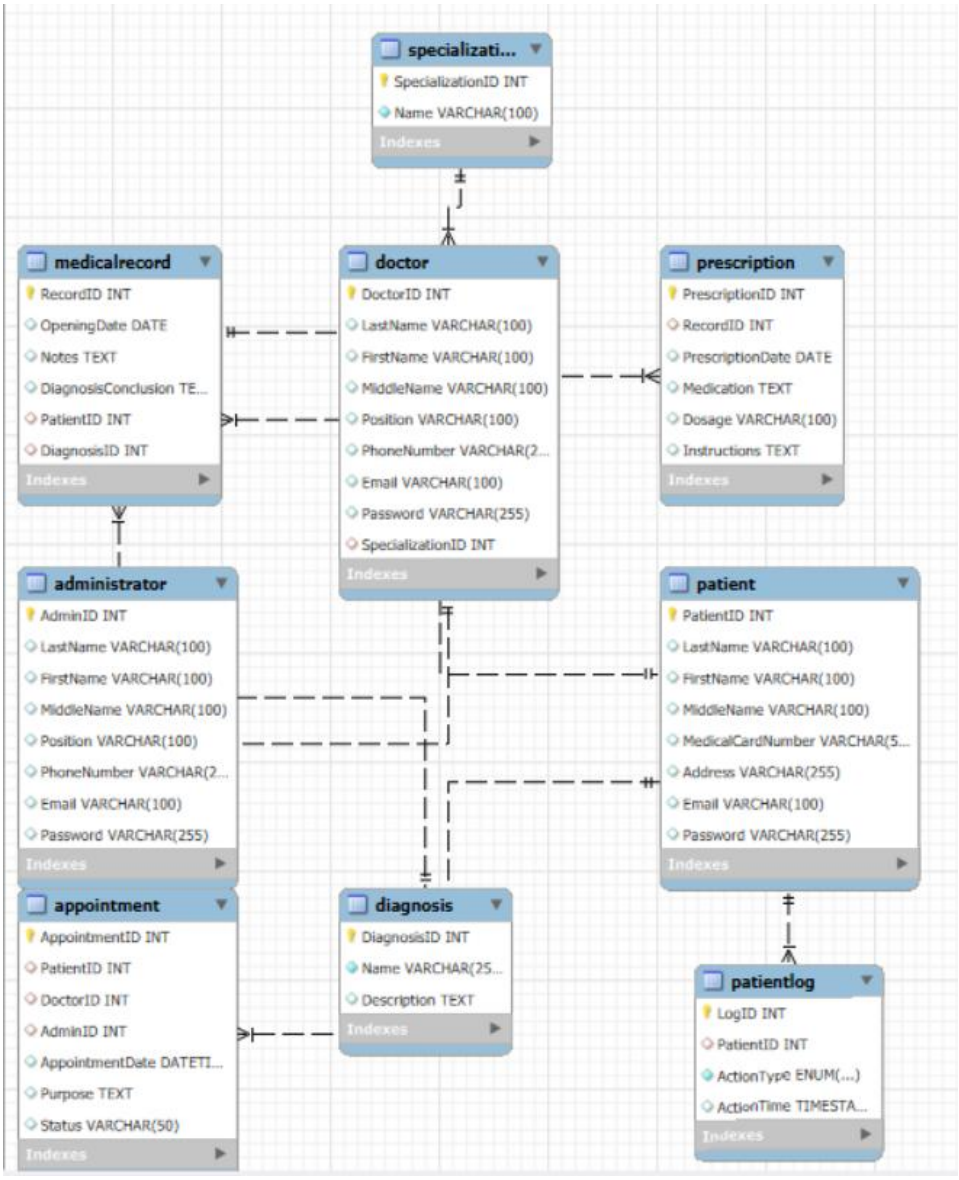


Рис.9 Створена у середовищі MySQL схема бази.

У процесі реалізації інформаційної системи Medix було використано MySQL як основну систему управління базами даних (СУБД), що забезпечує надійність, продуктивність та гнучкість при роботі з медичною інформацією.

SQL-запити для створення таблиць (зокрема patient, doctor, administrator, diagnosis, medicalrecord, prescription, patientlog) були написані вручну та виконувалися безпосередньо через MySQL Workbench або інші клієнти для адміністрування баз даних.

Усі таблиці мають чітко визначені первинні та зовнішні ключі, що гарантує логічну цілісність та дозволяє точно відтворити структуру предметної області. Було реалізовано необхідні обмеження (FOREIGN KEY, NOT NULL, UNIQUE) для забезпечення достовірності та актуальності даних.

Для керування обліковими записами користувачів реалізована власна логіка автентифікації, зокрема зберігання зашифрованих паролів, рольовий поділ доступу (пацієнт, лікар, адміністратор) та відповідні SQL-запити для перевірки прав.

Для керування доступом до медичних даних на логічному рівні реалізовано перевірку прав у запитах до бази даних. Наприклад, пацієнти мають доступ лише до власних записів, а лікарі — до тих, що пов’язані з ними через таблиці.

3 ПРИКЛАДНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Організаційна структура програмного забезпечення

Організаційна структура прикладного програмного забезпечення інформаційної системи управління медичними записами побудована за принципами багаторівневої архітектури, що забезпечує чітке розділення відповідальностей між основними частинами системи. Такий підхід спрощує розробку, тестування, масштабування та супровід програмного забезпечення.

Архітектура умовно поділена на три логічні рівні: Client, Logic, Database.

Рівень Client (клієнтська частина — Windows Forms) [6]

Цей рівень відповідає за графічний інтерфейс користувача та реалізований за допомогою WinForms. Він включає:

- Forms – вікна для реєстрації пацієнта, створення медичного запису, призначення діагнозів і рецептів. Забезпечують відображення полів вводу, кнопок, таблиць тощо;
- UserInteraction – обробляє дії користувача: натискання кнопок, введення даних, вибір пацієнта, ініціація збереження запису. Надсилає запити до логіки застосунку;
- Controls – багаторазові елементи інтерфейсу, такі як діалогові вікна, спливаючі повідомлення, таблиці для перегляду історії пацієнтів. Використовуються для побудови інтуїтивного UI.

Рівень Logic (логіка додатку)

Це центральний шар, що реалізує основну функціональність системи та містить:

- AuthModule – відповідає за авторизацію лікаря та адміністратора. Реалізує перевірку облікових даних та контроль доступу до функцій системи;
- PatientService – обробляє реєстрацію пацієнтів, зберігає їх дані, надає доступ до історії хвороби та дозволяє її оновлення;
- MedicalRecordService – керує створенням медичних записів, валідує дані, зв'язує записи з діагнозами та рецептами;
- DiagnosisService – дозволяє додавати нові діагнози або обирати наявні. Забезпечує пошук та редагування описів захворювань;
- PrescriptionService – реалізує функціональність призначення лікарських засобів, зберігає дозування, інструкції та тривалість прийому;
- AuditLogger – фіксує всі ключові дії користувачів (реєстрація, створення запису, авторизація), забезпечуючи контроль та прозорість.

Рівень Database [7] **(база даних — MySQL)**

Цей рівень забезпечує збереження та доступ до даних за допомогою СУБД MySQL. Він включає:

- DatabaseAccess – шар доступу до БД, що реалізує CRUD-операції через SQL-запити або ORM-бібліотеки. Спілкується безпосередньо з MySQL;
- DataModels – описує структури таблиць: patient, medicalrecord, diagnosis, prescription, patientlog. Моделі відповідають сутностям предметної області, визначають зв’язки між ними та обмеження даних;

Структурна схема взаємодії між цими компонентами наведена на рис.10.

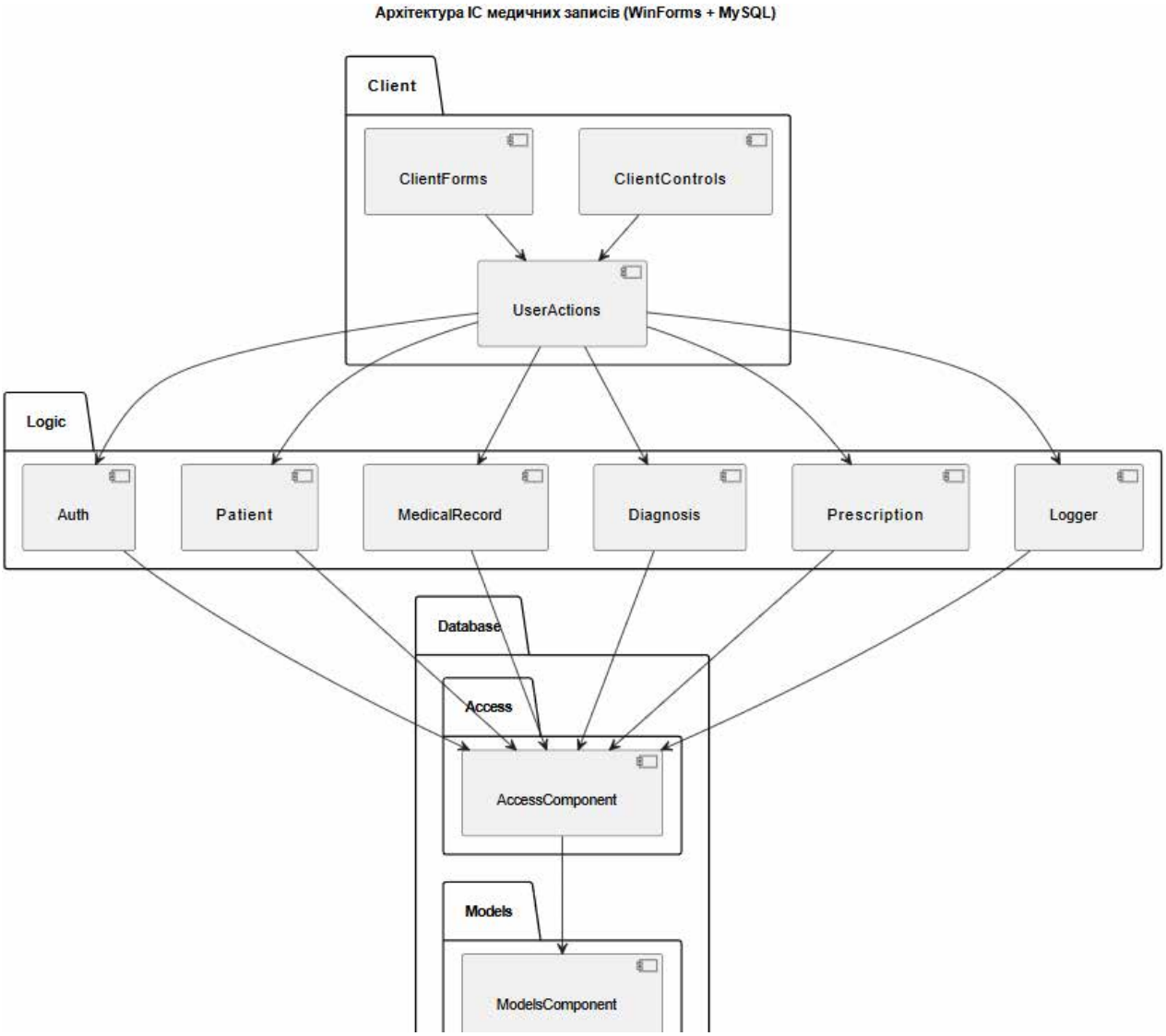


Рис.10 Діаграма пакетів інформаційної системи

Такий поділ на клієнтську частину, логіку та базу даних забезпечує ефективну масштабованість інформаційної системи медичних записів. Він створює чітку та логічну структуру, що спрощує тестування окремих компонентів, підвищує стабільність розробки та сприяє гнучкому розширенню функціоналу в майбутньому.

3.2 Вибір інструментарію для створення ППЗ

У процесі розробки інформаційної системи медичних записів було прийнято рішення використовувати перевірений, надійний та гнучкий набір інструментів, що дозволяє ефективно реалізувати функціональність, створити зручний графічний інтерфейс та забезпечити стабільну взаємодію з базою даних. Нижче наведено обґрунтування вибору основного інструментарію, використаного на всіх етапах створення програмного забезпечення.

Мова програмування: C#

C# було обрано [10] як основну мову розробки для даного проєкту завдяки її потужним об’єктно-орієнтованим можливостям, підтримці сучасних підходів до програмування (асинхронність, LINQ-запити, лямбда-вирази тощо) та широкому стандартному набору бібліотек. Вона ідеально підходить для створення десктопних застосунків у середовищі Windows і має тісну інтеграцію з .NET-платформою, що значно полегшує доступ до інструментів роботи з файлами, базами даних, мережевими запитами та іншим.

Середовище розробки: Visual Studio

Для реалізації програмного забезпечення використовувалося середовище Microsoft Visual Studio — один із найпотужніших IDE для C# та .NET. Visual Studio забезпечує:

- візуальний дизайнер форм для швидкої розробки інтерфейсу;
- налагодження, підказки IntelliSense, інтеграцію з системами керування версіями;
- підтримку підключення до СУБД, зокрема MySQL;
- зручне керування сторонніми бібліотеками через NuGet.

Технологія інтерфейсу: Windows Forms (WinForms)

Для створення графічного інтерфейсу користувача було обрано технологію Windows Forms, яка забезпечує просту, стабільну і перевірену реалізацію десктопних додатків. Основні переваги у контексті даного проєкту:

- швидке створення інтерфейсу через візуальний дизайнер;
- широкий набір стандартних елементів (Label, Button, TextBox, DataGridView тощо);
- інтеграція з .NET Framework і підтримка подійної моделі програмування;
- висока стабільність при запуску в середовищі Windows.

Попри певні обмеження (відсутність сучасних UI/UX механізмів, обмежене масштабування під DPI), WinForms повністю відповідає вимогам до функціональності й надійності медичної інформаційної системи.

База даних: MySQL

В якості системи управління базами даних обрано MySQL — реляційну СУБД з відкритим кодом, яка забезпечує високу продуктивність, підтримку транзакцій, зовнішніх ключів, індексів і перевірок цілісності. Вона добре інтегрується з .NET через відповідні драйвери (наприклад, MySql.Data) і дозволяє ефективно обробляти структури, характерні для медичних записів: пацієнти, діагнози, призначення, історія відвідувань тощо.

Додаткові бібліотеки та технології:

- MySql.Data — офіційний .NET-драйвер для взаємодії з MySQL;
- System.Text.Json / Newtonsoft.Json — для обробки конфігурацій і передачі структурованих даних;
- System.Net.Http — для підтримки мережевих запитів (у разі інтеграції зі сторонніми сервісами чи API);

- System.Text.RegularExpressions — для валідації введення користувача (наприклад, перевірка правильності номера медичної картки).

Таким чином, поєднання [9] C#, Windows Forms, Visual Studio та MySQL дозволило реалізувати повнофункціональну, зручну та надійну інформаційну систему для роботи з медичними записами. Обраний інструментарій забезпечив оптимальний баланс між швидкістю розробки, зручністю використання, продуктивністю та вимогами до безпеки й масштабованості.

3.3 Алгоритмізація та програмування програмних модулів

У процесі розробки програмного забезпечення «Medix» — інформаційної системи управління медичними записами — було застосовано модульний підхід до проєктування та програмування. Такий підхід передбачає поділ застосунку на ізольовані, взаємопов’язані програмні модулі з чітко визначеними зонами відповідальності. Це забезпечує гнучкість, зручність супроводу, тестованість і можливість масштабування системи.

Архітектурна модель

Система побудована на основі класичної трирівневої клієнт-серверної архітектури (рис.11), яка включає:

- рівень представлення (UI) — реалізований як Windows Forms-додаток, що забезпечує взаємодію користувача з системою. Дозволяє вводити, переглядати, редагувати медичні дані, а також здійснювати навігацію та управління процесами;
- рівень логіки доступу до даних — представлений модулем AccessComponent, [13] який інкапсулює всю логіку взаємодії з базою даних MySQL. Він забезпечує виконання SQL-запитів і обробку результатів згідно з вимогами бізнес-логіки;
- модельний рівень — набір класів-моделей, що описують основні сутності предметної області: пацієнти, лікарі, медичні записи, діагнози,

рецепти, адміністратори, тощо. Ці моделі забезпечують типізовану структуру даних і полегшують обробку запитів та відповідей між рівнями системи.

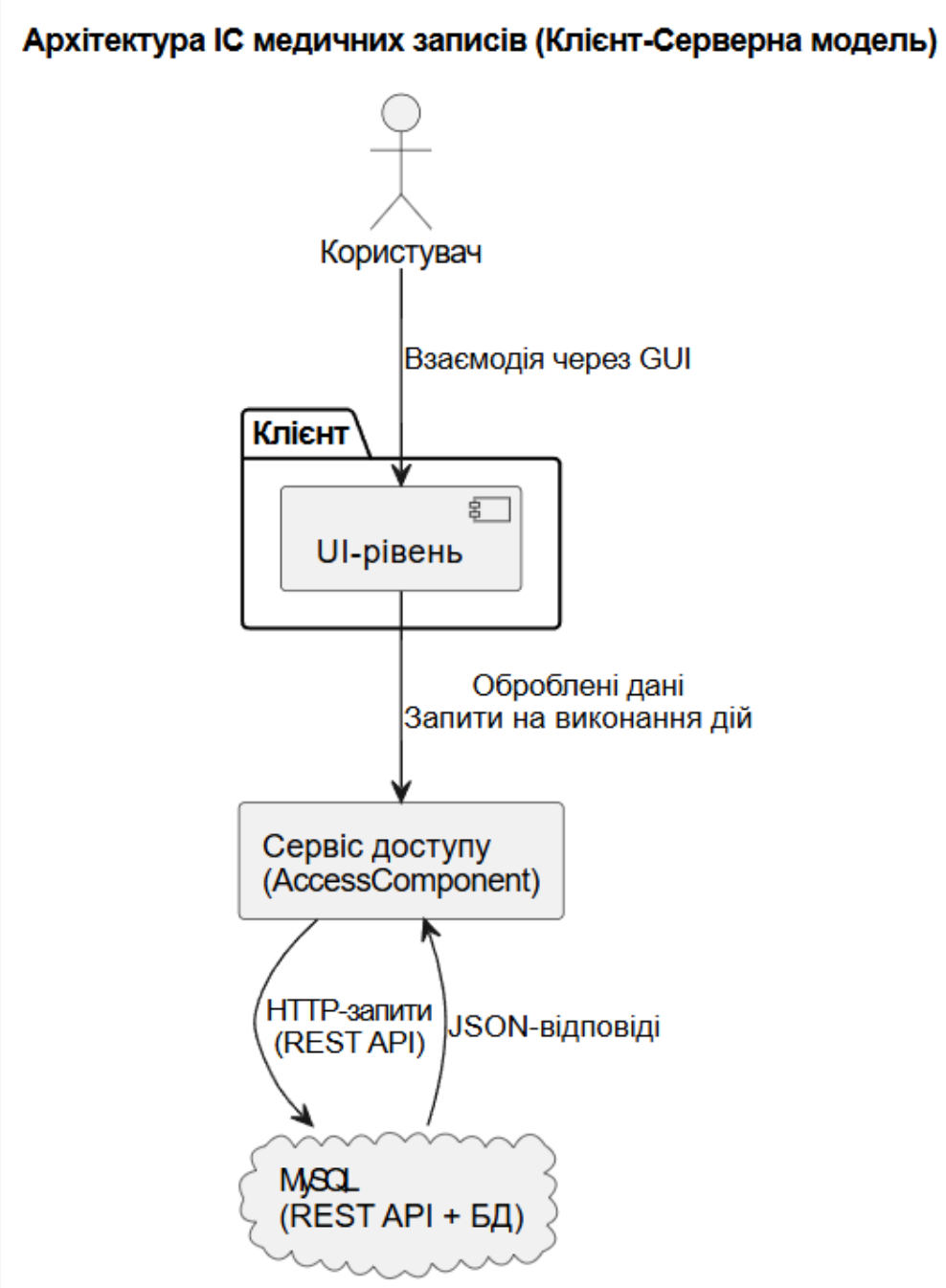


Рис.11 Архітектурна структура програми (Client–Server модель)

Розробка інтерфейсу користувача

Інтерфейс користувача [4] (англ. User Interface, UI) — це засіб ефективної взаємодії користувача з інформаційною системою. Він охоплює набір елементів для введення, обробки та відображення медичної інформації, адаптованих під потреби медичного персоналу (лікарів, адміністраторів тощо) для забезпечення комфорту та зручності у щоденній роботі.

У цьому розділі розглядається реалізація інтерфейсу (рис.12) для ключових форм додатку «Medix». Інтерфейс було створено із застосуванням технології Windows Forms (WinForms), що надала можливість реалізувати зрозумілу, функціональну та візуально доступну взаємодію з інформаційною системою.

Проєктування UI базувалося на принципах:

- простоти — мінімум зайвих елементів, лише необхідний функціонал;
- зручності — логічна структура форм і зрозуміла навігація;
- наочності — чітке візуальне оформлення елементів керування, таких як кнопки, поля введення, списки пацієнтів і діагнозів.

Навігація в системі здійснюється через меню, вкладки та кнопки з чіткими позначеннями, що спрощує доступ до функцій, таких як реєстрація пацієнтів, створення медичних записів, додавання діагнозів і призначення рецептів.

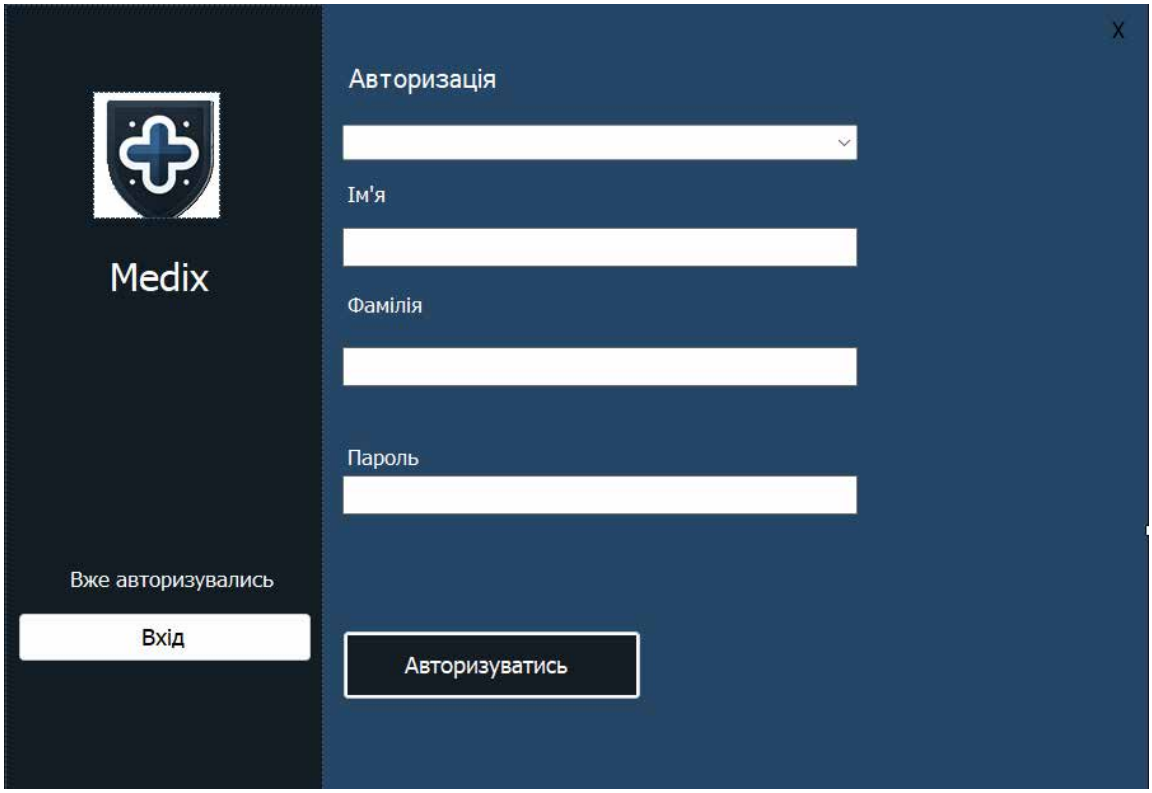


Рис.12.1 Вікно авторизації в системі

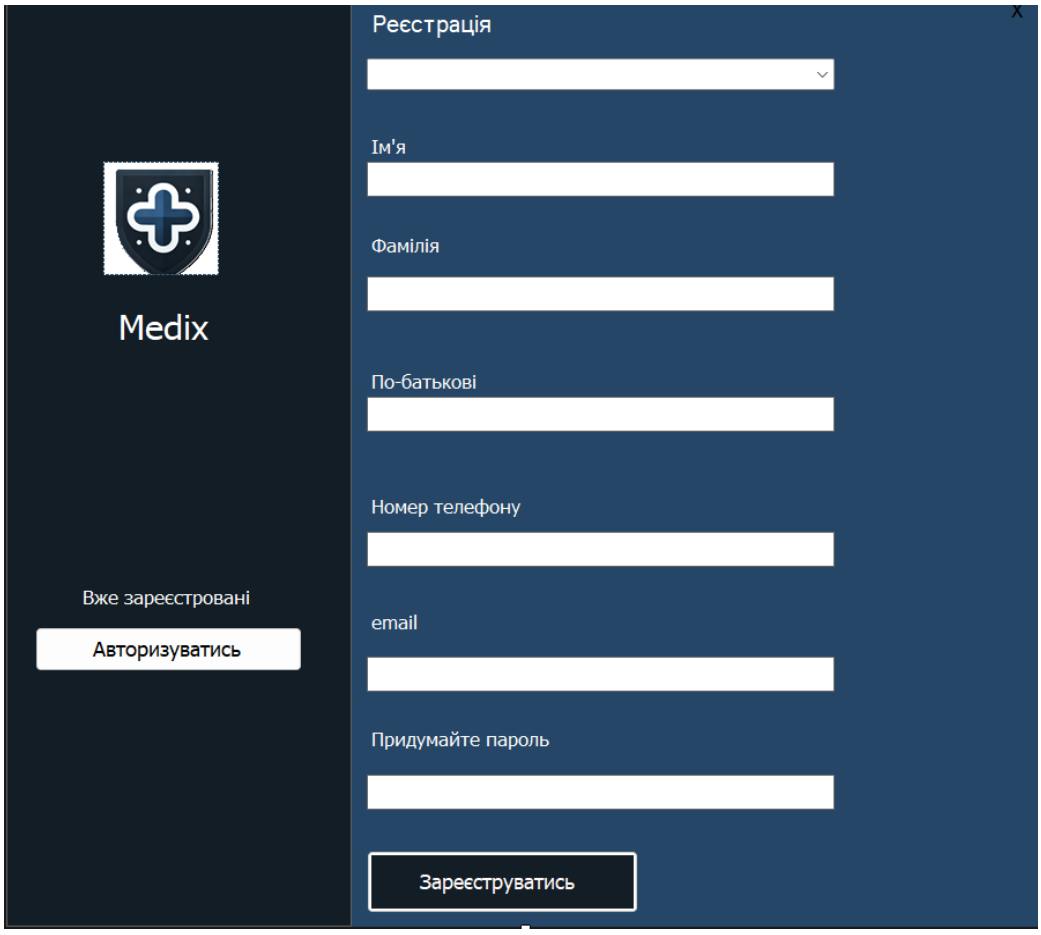


Рис.12.2 Вікно реєстрації

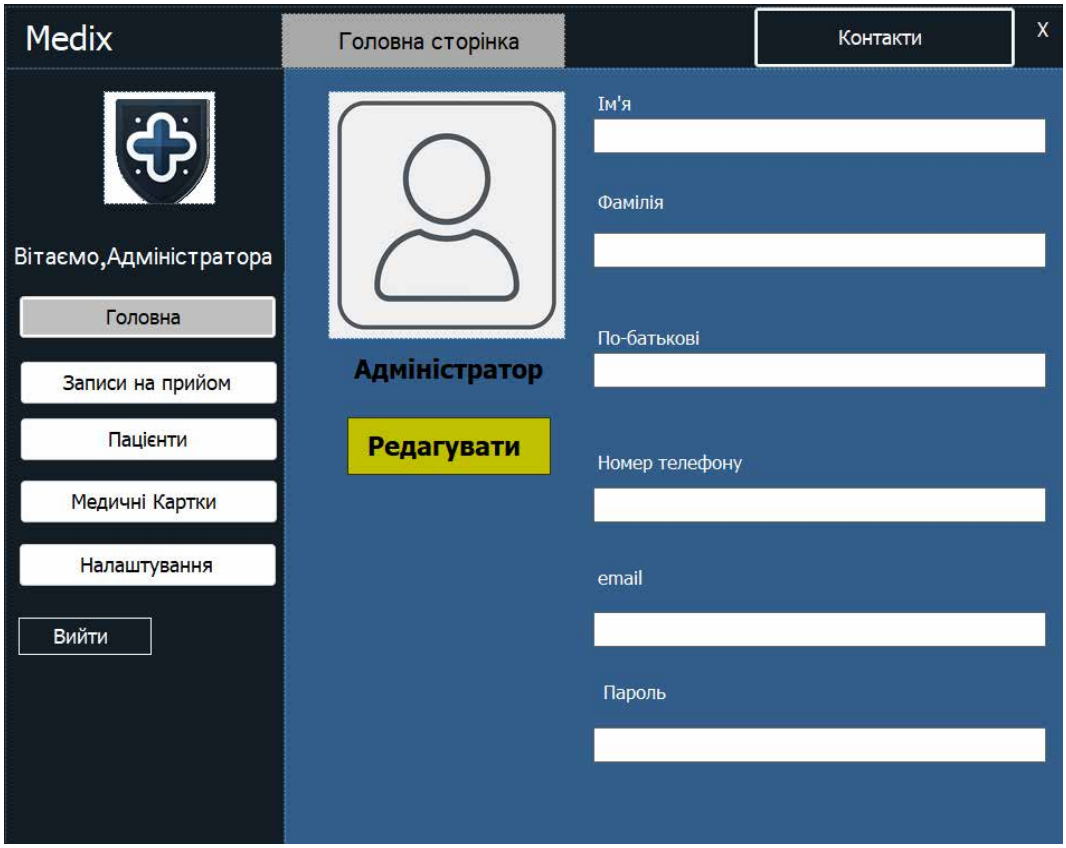


Рис.12.3 Вікно головної сторінки адміністратора

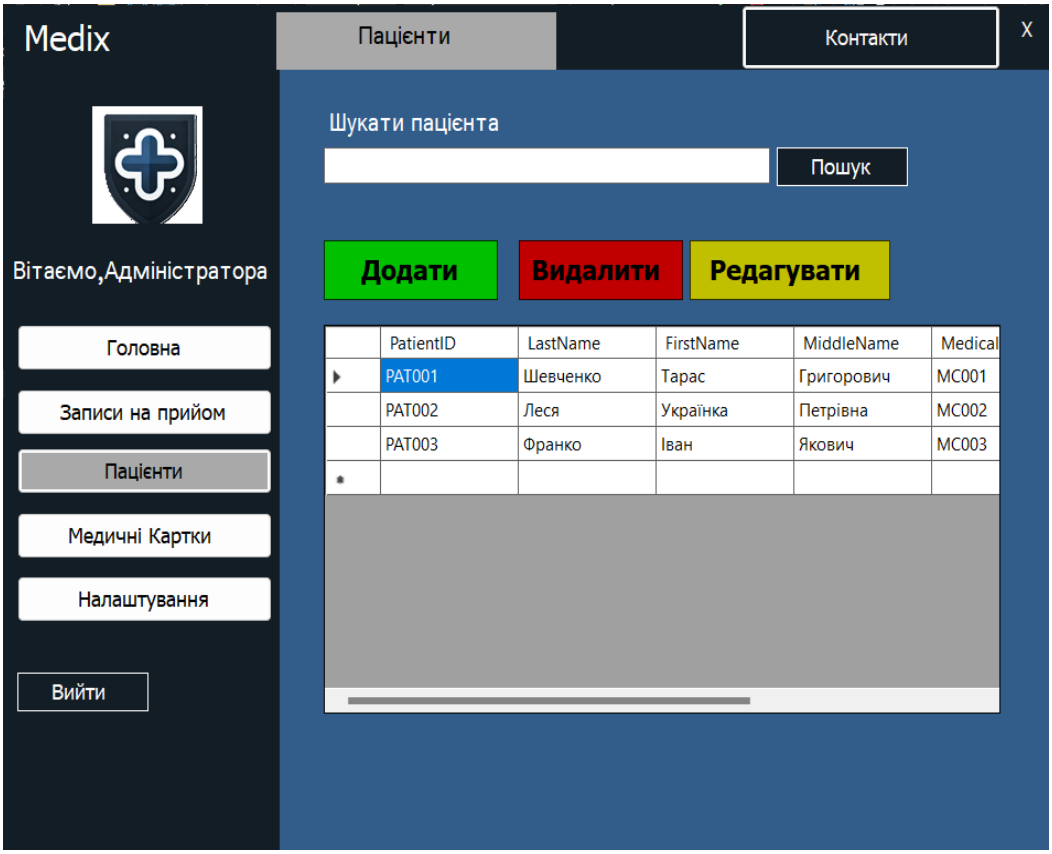


Рис.12.4 Вікно пацієнтів адміністратор

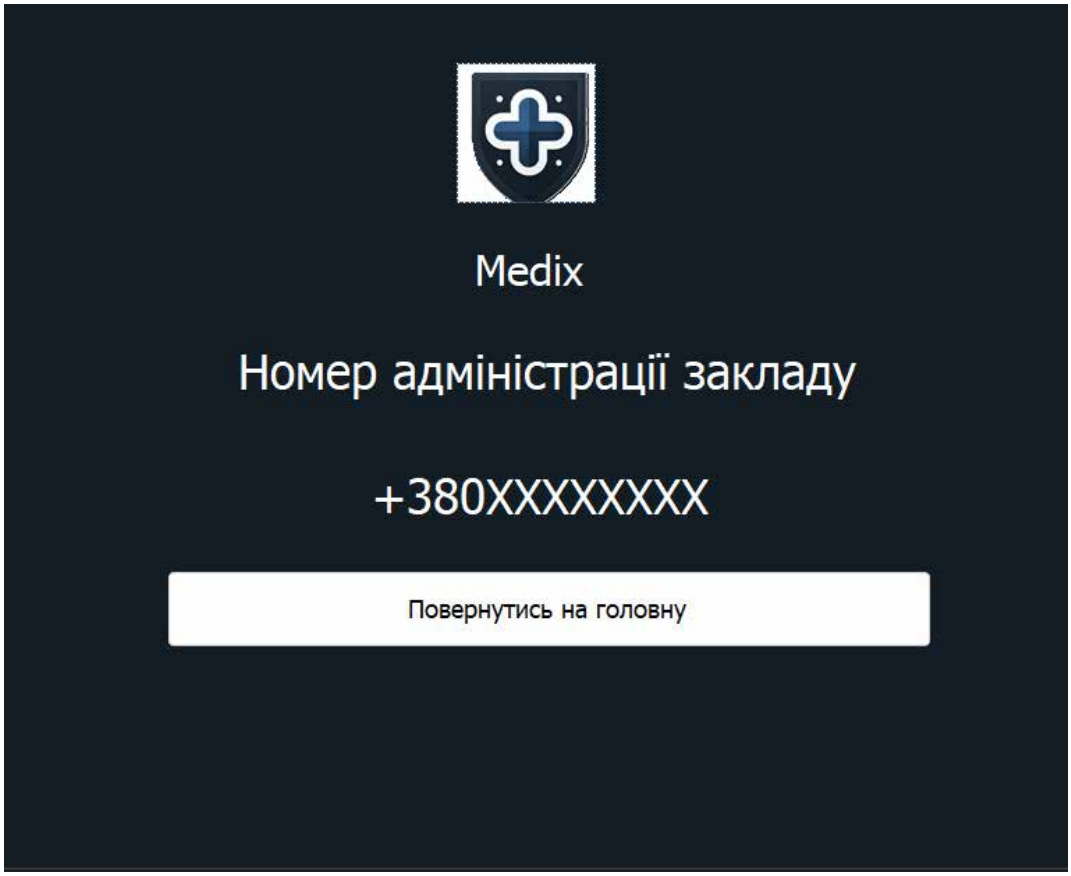


Рис.12.5 Вікно контактів

Організація підключення до бази даних

Для взаємодії з базою даних, розгорнутою на платформі MySQL, у системі «Medix» реалізовано підключення на основі [7] MySqlConnection. Основна конфігурація здійснюється безпосередньо у формі, де реалізовано завантаження та обробку даних. Організація підключення до бази даних наведено на рис. 13

```
private string connectionString =
    "server=localhost;port=3306;database=medical_clinic;uid=root;pwd=Unastya0987978273;";


private void LoadPatientData()
{
    using (MySqlConnection connection = new MySqlConnection(connectionString))
    {
        try
        {
            connection.Open();
            string query = "SELECT * FROM patient";
            MySqlDataAdapter adapter = new MySqlDataAdapter(query, connection);
            DataTable table = new DataTable();
            adapter.Fill(table);
            dataGridViewPatient.DataSource = table;
        }
        catch (Exception ex)
        {
            MessageBox.Show("Помилка при завантаженні даних: " + ex.Message);
        }
    }
```

Рис. 13 Організація підключення до бази даних на C#

Такий підхід є логічно обґрунтованим і технічно доцільним з кількох причин:

- цілісність та локалізація логіки — усі дії з підключення та завантаження даних реалізовано в одному методі, що забезпечує простоту контролю й розуміння логіки;
- уніфікація підключення — використання єдиного рядка підключення дозволяє уникати дублювання параметрів у різних модулях;

- гнучкість та масштабованість — структура дозволяє легко адаптувати запити або перенести логіку в окремий сервіс-клас;
- підвищення тестованості — метод легко виділяється в окремий клас для подальшого розширення або модульного тестування.

Таким чином, обраний підхід дозволяє підтримувати чисту, зрозумілу архітектуру взаємодії з MySQL-базою в контексті системи «Medix», забезпечуючи надійність, підтримуваність та відповідність сучасним принципам програмування.

Централізована логіка доступу до медичних даних

У системі «Medix» доступ до даних здійснюється за допомогою SQL-запитів через інтерфейс MySqlConnection. Цей підхід дає змогу реалізувати повноцінну та структуровану роботу з усіма медичними даними системи.

Функціональність охоплює:

- авторизацію користувачів (лікарі та адміністратори) з урахуванням ролей та прав доступу;
- створення, редагування та видалення основних сутностей: пацієнтів, медичних записів, діагнозів, рецептів;
- фільтрацію та пошук пацієнтів і записів за ПІБ, датою прийому, діагнозом тощо;
- формування звітів — виведення агрегованих даних, наприклад, кількості прийомів або найчастіших діагнозів;
- контроль доступу до функцій системи на основі ролі користувача (адміністратор / лікар).

Частину логіки взаємодії з базою даних можна розглянути на прикладі реалізації завантаження інформації про пацієнтів із таблиці patient на рис.14.

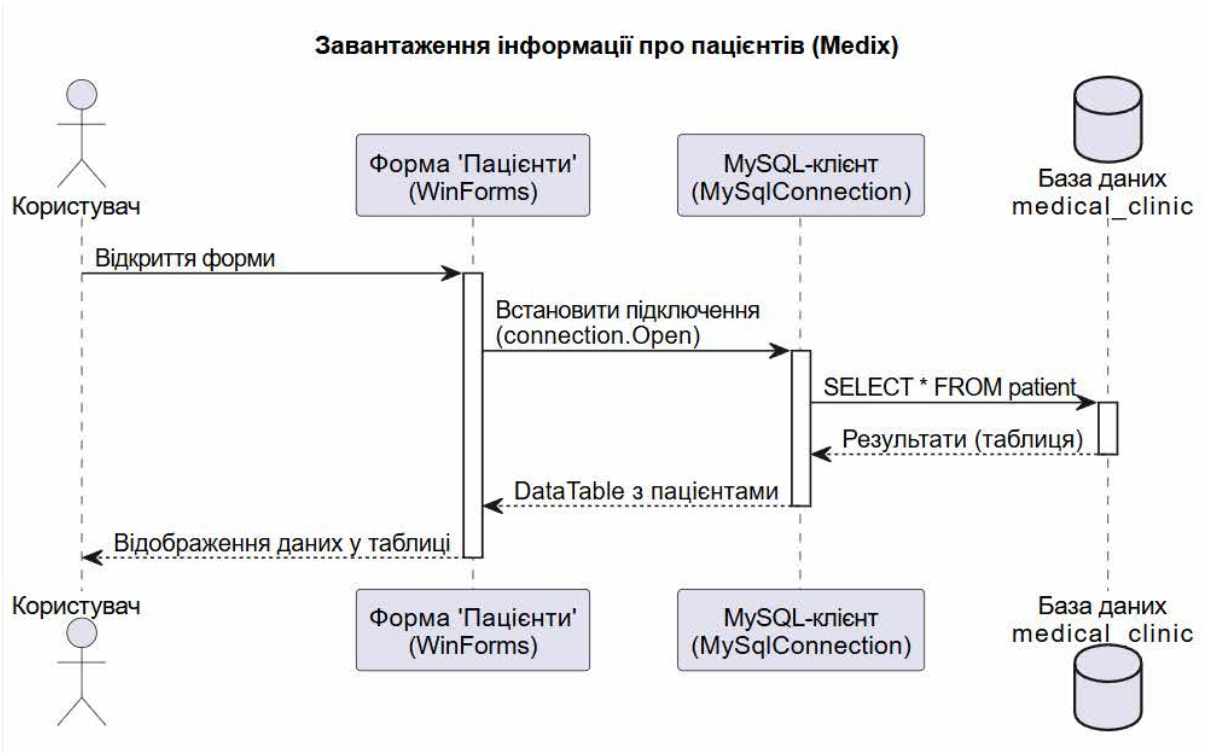


Рис.14 Алгоритм взаємодії клієнтської форми з базою даних

На рис.14 відображено поетапну логіку взаємодії з базою даних під час додавання нового медичного запису. Процес ініціюється після натискання кнопки «Додати запис» у відповідній формі користувацького інтерфейсу, після чого виконується така послідовність дій:

- перевірка валідності полів введення: прізвище пацієнта, обраний діагноз, дані про рецепт, дата відкриття запису тощо;
- вставка нового запису до таблиці medicalrecord — зберігається дата створення, примітки, зв’язок із пацієнтом;
- додавання нового діагнозу до таблиці diagnosis, якщо він ще не існує у базі (перевірка унікальності);
- створення рецепта у таблиці prescription із зазначенням медикаментів, дозування та інструкцій;
- формування зв’язків між записом і пов’язаними сутностями: PatientID, DiagnosisID, PrescriptionID;

- у разі виникнення помилки на будь-якому з кроків — користувачу виводиться відповідне повідомлення, а подальше виконання зупиняється.

Методи взаємодії з MySQL [7] **для додавання нового медичного запису**

Реалізація здійснена на основі MySqlConnection у C# наведено на рис. 15.

1. **Метод InsertDiagnosis** — додає новий діагноз у таблицю diagnosis (якщо його ще немає):

```
public bool InsertDiagnosis(string name, string description)
{
    using (var connection = new MySqlConnection(connectionString))
    {
        connection.Open();
        string query = "INSERT INTO diagnosis (Name, Description) VALUES (@name, @description)";
        var command = new MySqlCommand(query, connection);
        command.Parameters.AddWithValue("@name", name);
        command.Parameters.AddWithValue("@description", description);
        return command.ExecuteNonQuery() > 0;
    }
}
```

Рис.15.1 Метод InsertDiagnosis

2. **Метод InsertPrescription** — створює новий запис у таблиці prescription:

```
public bool InsertPrescription(DateTime date, string medication, string dosage, string instructions)
{
    using (var connection = new MySqlConnection(connectionString))
    {
        connection.Open();
        string query = "INSERT INTO prescription (PrescriptionDate, Medication, Dosage, Instructions) " +
                       "VALUES (@date, @med, @dose, @instr)";
        var command = new MySqlCommand(query, connection);
        command.Parameters.AddWithValue("@date", date);
        command.Parameters.AddWithValue("@med", medication);
        command.Parameters.AddWithValue("@dose", dosage);
        command.Parameters.AddWithValue("@instr", instructions);
        return command.ExecuteNonQuery() > 0;
    }
}
```

Рис.15.2 Метод InsertPrescription

3. **Метод InsertMedicalRecord** — додає запис у таблицю medicalrecord, зв’язуючи пацієнта, діагноз і рецепт:

```
public bool InsertMedicalRecord(string patientId, string diagnosisId, string prescriptionId, string notes, DateTime openingDate)
{
    using (var connection = new MySqlConnection(connectionString))
    {
        connection.Open();
        string query = "INSERT INTO medicalrecord (PatientID, DiagnosisID, PrescriptionID, Notes, OpeningDate) " +
                       "VALUES (@patient, @diagnosis, @prescription, @notes, @date)";
        var command = new MySqlCommand(query, connection);
        command.Parameters.AddWithValue("@patient", patientId);
        command.Parameters.AddWithValue("@diagnosis", diagnosisId);
        command.Parameters.AddWithValue("@prescription", prescriptionId);
        command.Parameters.AddWithValue("@notes", notes);
        command.Parameters.AddWithValue("@date", openingDate);
        return command.ExecuteNonQuery() > 0;
    }
}
```

Рис.15.3 Метод InsertMedicalRecord

Клас доступу до бази даних у системі «Medix» виступає централізованим модулем для обробки всіх запитів до MySQL. Такий підхід:

- забезпечує модульність та повторне використання коду;
- дає змогу інкапсулювати логіку запитів, не розміщуючи SQL безпосередньо у формах;
- підвищує тестованість і підтримуваність коду;
- підходить для розширення функціоналу та масштабування системи (наприклад, додавання логів, журналів дій, перевірок прав доступу тощо).

Моделі даних (Patient, Doctor, MedicalRecord тощо)

Кожна сутність у системі «Medix» реалізована як клас-модель. У моделі використовуються анотації типу [JsonProperty] для зручного зв’язку зі структурою бази даних та, за потреби, автоматичної десеріалізації JSON-об’єктів, отриманих у результаті запитів.

Це дозволяє працювати з медичними даними у типізованому вигляді, знижуючи ризик помилок при ручній обробці запитів та полегшуючи інтеграцію з інтерфейсом користувача.

Приклад: модель Patient

Модель Patient містить основні поля, які відповідають структурі таблиці patient у базі даних medical_clinic наведено на рис. 16.

```
public class Patient
{
    [JsonProperty("patientid")]
    public string PatientID { get; set; }
    [JsonProperty("lastname")]
    public string LastName { get; set; }
    [JsonProperty("firstname")]
    public string FirstName { get; set; }
    [JsonProperty("middlename")]
    public string MiddleName { get; set; }
    [JsonProperty("medicalcardnumber")]
    public string MedicalCardNumber { get; set; }
    [JsonProperty("address")]
    public string Address { get; set; }
    [JsonProperty("email")]
    public string Email { get; set; }
    [JsonProperty("phonenumber")]
    public string PhoneNumber { get; set; }
}
```

Рис. 16 модель Patient

Усі запити до бази виконуються в асинхронному режимі, що забезпечує відгукливість інтерфейсу та стабільну роботу програми навіть при великому обсязі даних. Такий підхід дозволяє:

- централізовано керувати логікою взаємодії з базою даних;
- ефективно структурувати дані згідно з моделлю предметної області;
- підтримувати масштабовану та підтримувану архітектуру системи.

Це робить «Medix» готовим до подальшого розвитку: розширення функціоналу, реалізації звітів, захищеного доступу й хмарної інтеграції.

4 РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ СИСТЕМИ

4.1 Тестування системи

Завершальним етапом розробки системи «Medix» стало тестування, метою якого було перевірити функціональність, надійність і відповідність програмного забезпечення поставленим вимогам. У рамках цього етапу було проведено комплексне тестування системи, яке включало:

- функціональне тестування основних сценаріїв користування;
- перевірку коректності обробки введених даних;
- обробку виняткових ситуацій та помилок введення;
- оцінку зручності та зрозумілості інтерфейсу користувача (UI);
- базову перевірку безпеки та продуктивності при роботі з базою даних.

Тестування функціональності

Було протестовано роботу основних функцій системи яку наведено на рисунку 17, згідно з технічним завданням і функціональними вимогами. Зокрема:

Реєстрація та авторизація користувачів:

- для створення нового користувача (адміністратора або лікаря) потрібно заповнити всі обов’язкові поля відповідної форми;
- у разі правильного введення даних (ім’я, прізвище, логін, пароль тощо) — система успішно створює обліковий запис;
- у разі порушення правил валідації (наприклад, некоректний формат email або порожнє поле) — відображається повідомлення з описом помилки;
- для реєстрації адміністратора передбачено введення спеціального службового пароля, який не зберігається у відкритому вигляді та передається замовнику окремо разом з інсталяційним пакетом.

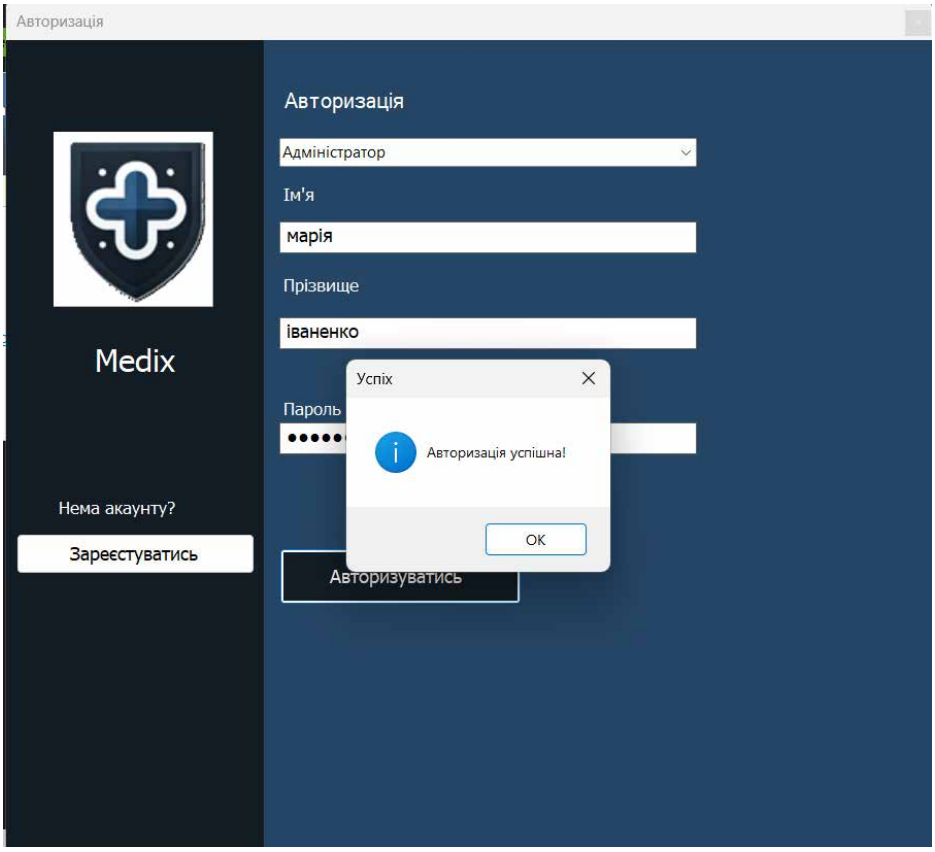


Рис.17.1 Авторизація в системі

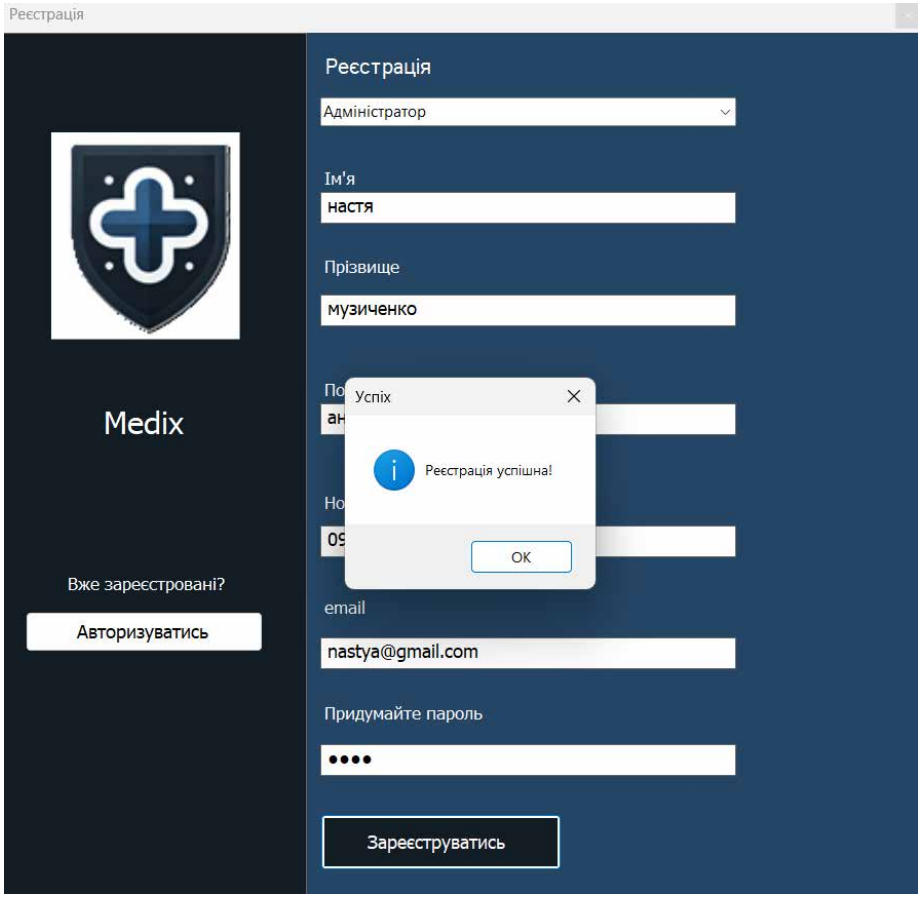


Рис.17.2 Реєстрація користувача в ролі адміністратора

У системі реалізовано функціонал реєстрації та авторизації користувачів. Під час реєстрації користувач вводить особисті дані, такі як ім’я, прізвище, посада, номер телефону, email та пароль. Після успішної реєстрації виводиться повідомлення про успіх. Увійти до системи можна через форму авторизації, де необхідно вказати відповідну роль, ім’я, прізвище та пароль. У випадку правильного введення даних система сповіщає про успішну авторизацію.

Додавання, редагування та видалення медичних записів — функціонал, доступний лише користувачам із правами адміністратора. Меню керування включає декілька вкладок, які відповідають за адміністрування електронних медичних карток, пацієнтів, лікарів, діагнозів, призначень тощо.

Для прикладу, розглянемо процес **додавання нового медичного запису** до системи (рис.18). Адміністратор переходить до відповідної вкладки, заповнює всі обов’язкові поля (наприклад, ПІБ пацієнта, дату, опис симптомів, діагноз та призначене лікування) та зберігає дані. Після успішного збереження система повідомляє про успішне додавання запису.

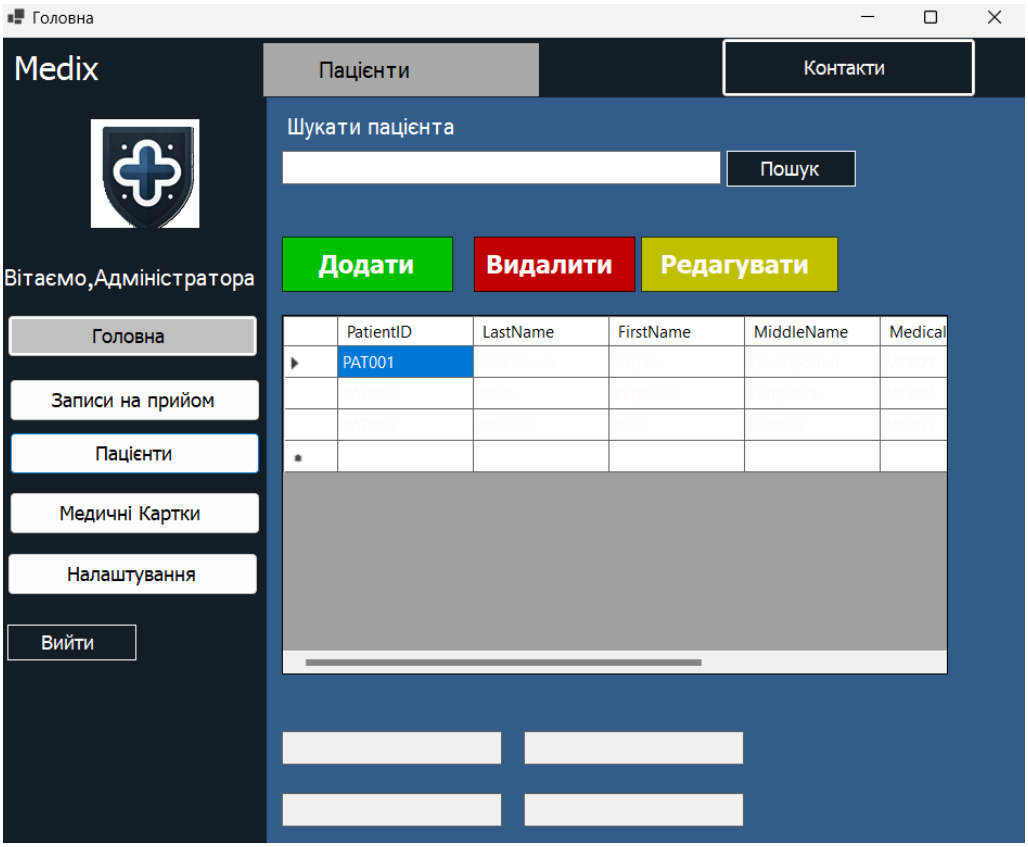


Рис.18.1 Вікно керування пацієнтами

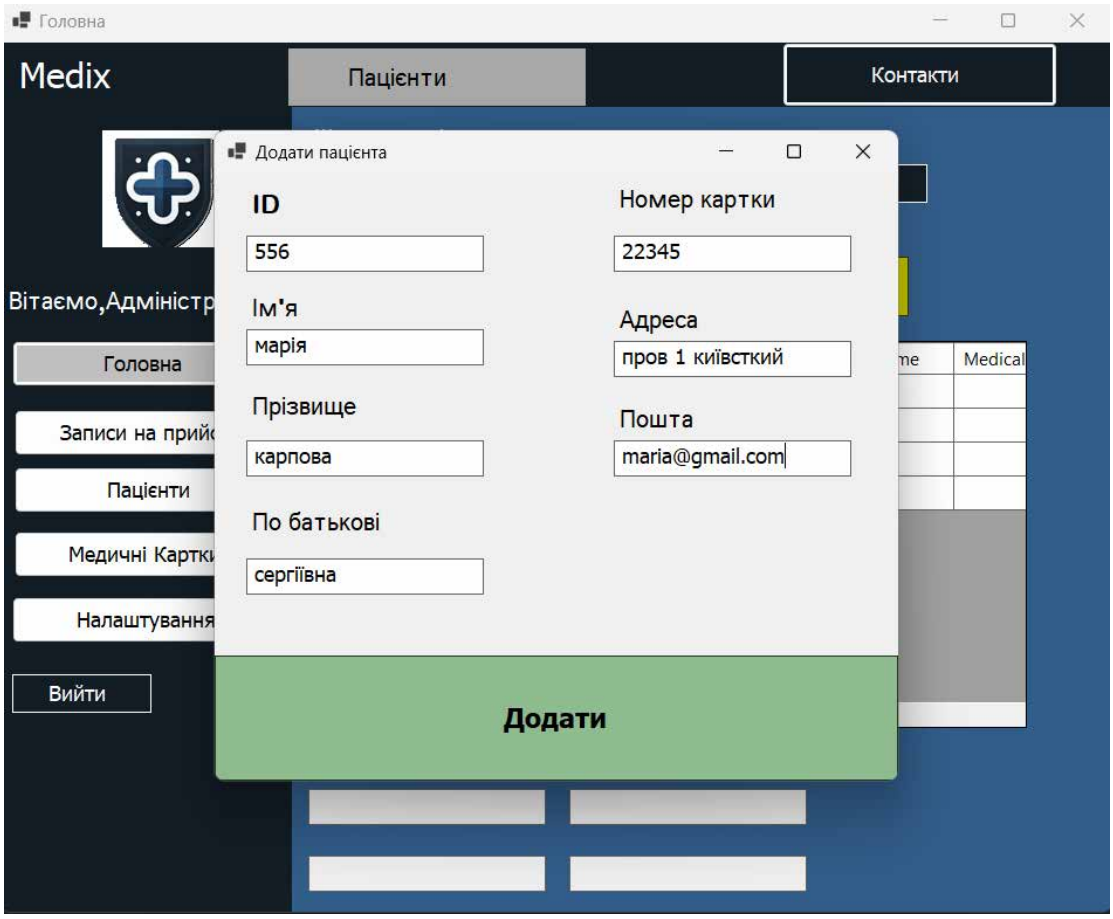


Рис.18.2 Внесення усіх метаданих пацієнта

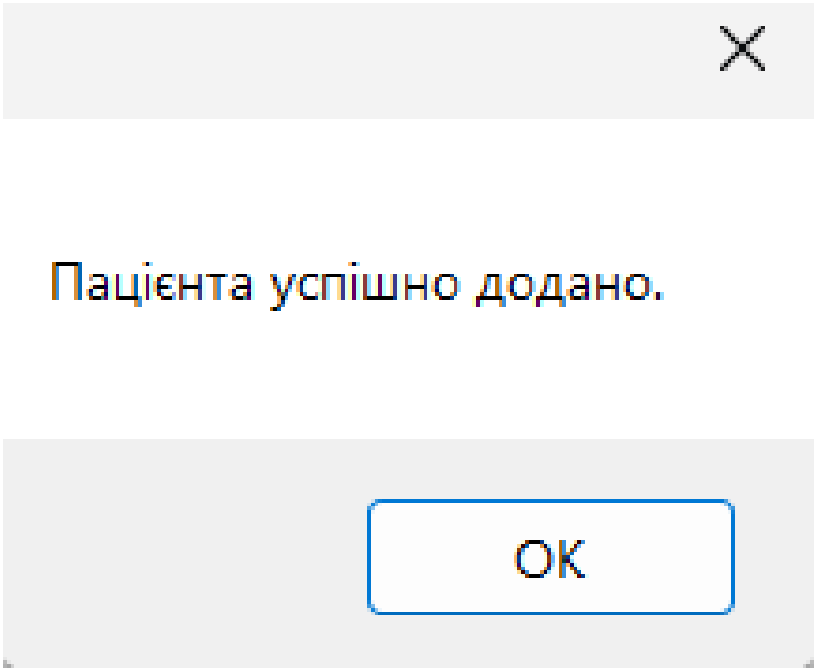


Рис.18.3 Повідомлення про успішне додавання

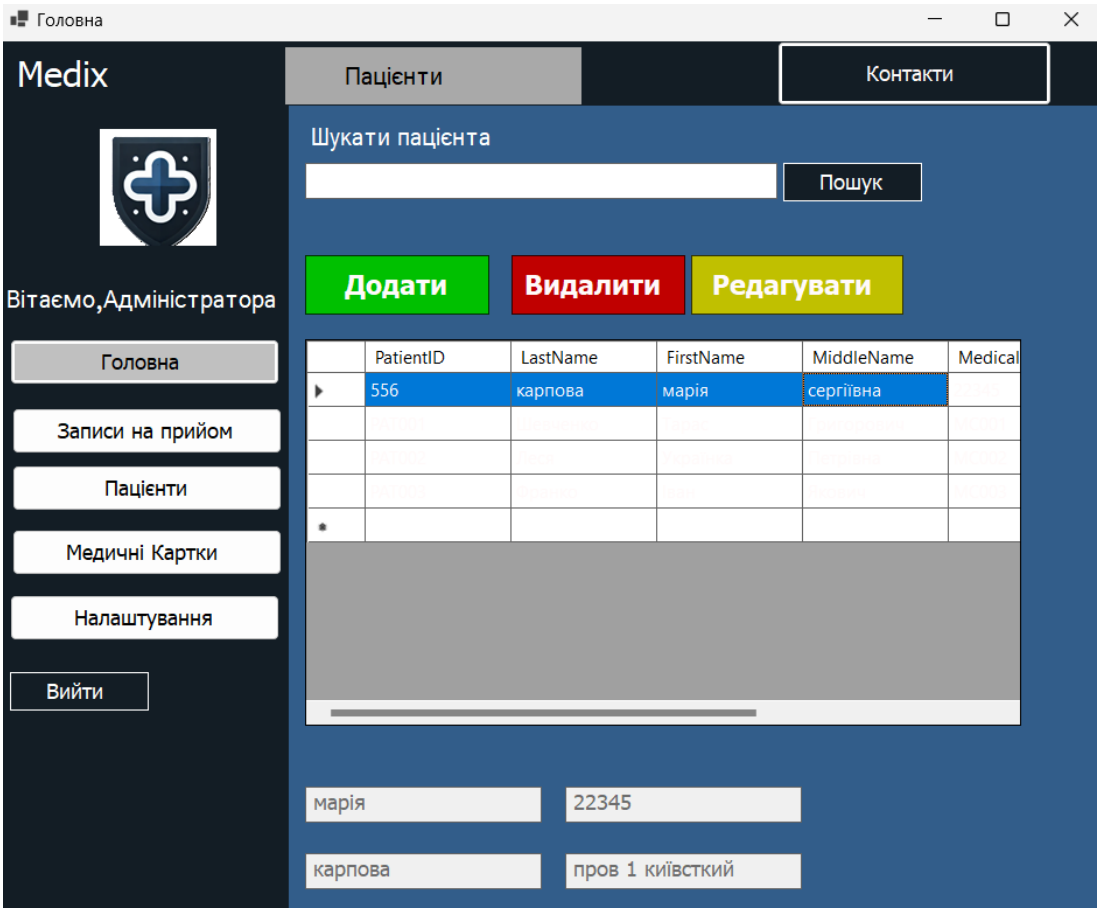


Рис.18.4 Поява нового пацієнта

Система правильно обробляє інформацію, не порушуючи загальну стабільність, і забезпечує інформативне повідомлення для користувача.

Тестування інтерфейсу користувача [4]

Інтерфейс системи управління медичними записами було протестовано на відповідність основним принципам зручності та юзабіліті, зокрема:

- логічна структура розділів: всі функції згруповані у зрозумілі вкладки (пацієнти, лікарі, медичні записи, діагнози тощо);
- інтуїтивно зрозумілий та мінімалістичний дизайн: дозволяє швидко орієнтуватися у системі без додаткового навчання;
- реакція на дії користувача: система забезпечує миттєвий зворотний зв’язок (наприклад, повідомлення про успішну авторизацію, реєстрацію або додавання медичного запису);

- адаптивність інтерфейсу: коректне відображення елементів на різних розмірах екранів, включно з ноутбуками, моніторами та планшетами.

Результати тестування підтвердили, що інтерфейс є інтуїтивно зрозумілим, а навігація системою — логічною та зручною для користувачів різних ролей (адміністраторів, лікарів,).

4.2 Вимоги до апаратного та програмного забезпечення

Для забезпечення ефективного функціонування інформаційної системи «Medix» визначено вимоги до апаратного та програмного забезпечення з урахуванням архітектурної моделі, функціонального призначення системи та передбачуваного навантаження.

З огляду на використання трирівневої клієнт-серверної архітектури, система складається з двох основних компонентів:

Клієнтський вузол — робоча станція лікаря або адміністратора, де встановлено локальний [6] WinForms-додаток (Medix.exe).

Серверна частина — MySQL-сервер, який зберігає всю медичну інформацію, обробляє SQL-запити та забезпечує централізований доступ до бази даних.

Діаграма розгортання наведено на рисунку 19.

На діаграмі рознортання зображено логічну структуру системи, що включає такі основні компоненти:

Клієнтський додаток Medix (локальний WinForms-додаток), через який лікарі й адміністратори здійснюють введення, перегляд і редагування даних пацієнтів, медичних записів, діагнозів і рецептів.

Сервер бази даних MySQL, що розміщується у локальній мережі або на віддаленому сервері, до якого підключається клієнтська частина через TCP/IP з'єднання (порт 3306).

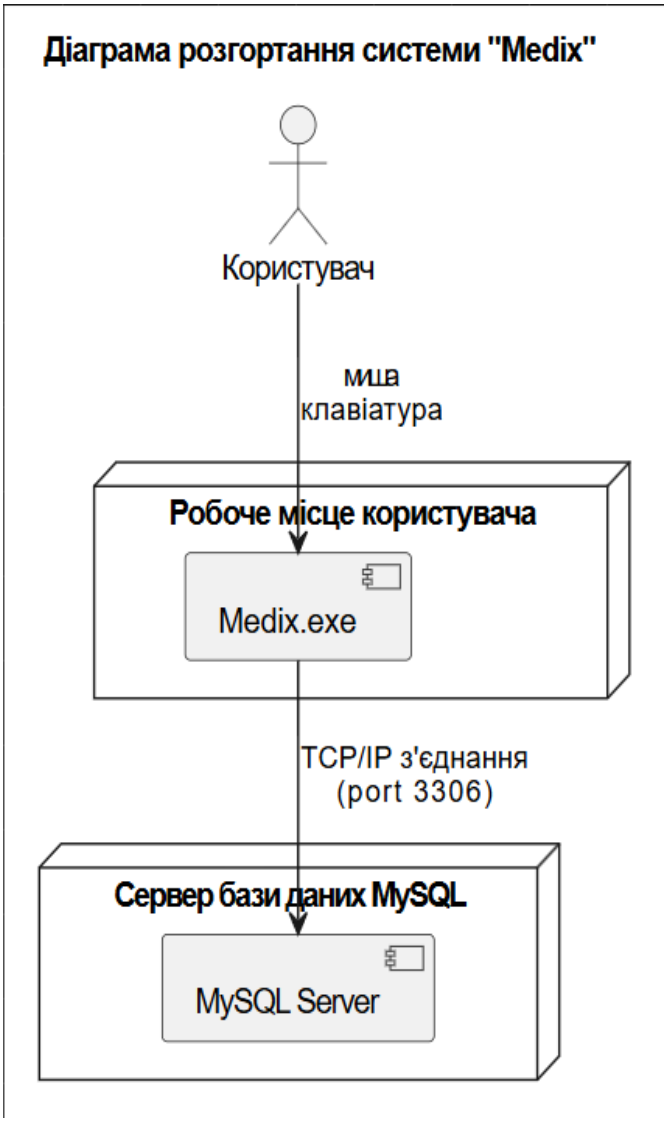


Рис.19 Діаграма розгортання компонентів системи

Таблиця 2

Апаратні вимоги клієнтської частини

Компонент	**Мінімальні вимоги**	**Рекомендовані вимоги**
Процесор (CPU)	2-ядерний, 2.0 GHz	4-ядерний, 2.5 GHz або вищий
Оперативна пам’ять	4 ГБ	8 ГБ або більше
Відеокарта	Інтегрована	Дискретна або інтегрована з підтримкою DirectX 11
Вільне місце на диску	500 МБ	1 ГБ або більше

Мережеве підключення	Локальна мережа або інтернет (мін. 5 Мбіт/с)	Стабільне підключення до серверу бази даних

Таблиця 3

Програмні вимоги клієнтської частини

Програмне забезпечення	**Мінімальна версія**	**Примітка**
Операційна система	Windows 10 або новіша	Рекомендується Windows 11
.NET Framework	.NET 6.0 або новіша	Потрібна для запуску WinForms-додатку
Драйвер доступу до БД	MySQL Connector/NET	Для взаємодії з MySQL (ADO.NET або ORM)
СУБД (для серверної частини)	MySQL 8.0 або новіша	Сервер бази даних для зберігання медичних записів
Антивірус/фаєрвол	Необов’язково	Має дозволяти вихідні з’єднання до MySQL-сервера
Браузер (опціонально)	Google Chrome, Edge, Firefox	Для завантаження оновлень або перегляду звітів

Програмно-архітектурні особливості серверної частини:

- СУБД: MySQL 8.0+
- Доступ до даних: ADO.NET або ORM (на розсуд розробника)
- API (опціонально): У разі потреби — можлива інтеграція REST API між сервером та додатком

4.3 Склад інсталяційного пакету

Інсталяційний пакет інформаційної системи «Medix» забезпечує автоматизовану установку всіх необхідних компонентів клієнтської частини на робочих станціях лікарів і адміністраторів.

Відповідно до архітектурної моделі (див. рис.19), серверна частина бази даних MySQL розгортається окремо — на локальному сервері або віддаленому хостингу, і не потребує локальної інсталяції. Клієнтська частина функціонує як десктоп-застосунок, що підключається до бази даних через TCP/IP-з’єднання.

Таблиця 4

Склад інсталяційного пакету системи «Medix»

Назва файлу/папки	Тип	Опис
Medix.Setup.msi	Інсталятор	Основний інсталяційний файл, створений за допомогою WiX або Inno Setup.
Medix.exe	Виконуваний файл	Основний клієнтський WinForms-додаток для взаємодії з системою.
MySql.Data.dll	Бібліотека	Драйвер для підключення до MySQL-серверу через ADO.NET.
System.Configuration.dll	Системна бібліотека	Обробка конфігураційних параметрів, зокрема connection string.
appsettings.json	Конфігурація	Налаштування для з’єднання з базою даних (сервер, порт, БД, логін).
README.txt	Документація	Інструкція з встановлення, налаштування та запуску додатку.
LICENSE.txt	Ліцензія	Текстова ліцензія на використання програмного забезпечення.
Uninstall.exe	Деінсталятор	Програма повного видалення додатку та очищення пов’язаних файлів.
Logs/	Папка	Каталог для збереження лог-файлів (створюється автоматично).
Certificates/ (опц.)	Папка	SSL-сертифікати для безпечного підключення до сервера (якщо потрібно).

Процес інсталяції

1. **Запуск інсталятора (MSI).** Подвійний клік на Medix.Setup.msi запускає процес. Перевіряється наявність .NET 6.0 або новішої версії. При потребі пропонується встановлення.
2. **Копіювання файлів.** Усі компоненти копіюються до обраного каталогу (типово: C:\Program Files\Medix\).
3. **Налаштування конфігурації.** Під час встановлення користувач вводить або підтверджує параметри підключення до MySQL-сервера: адресу, порт, логін, пароль.
4. **Створення ярликів.** Автоматичне створення ярлика на робочому столі та у меню «Пуск».
5. **Завершення.** Інсталятор пропонує відразу запустити додаток «Medix».

ВИСНОВКИ

Дана бакалаврська кваліфікаційна робота присвячена розробці інформаційної системи для управління медичними записами «Medix». Метою проєкту було створення зручного, ефективного та функціонального застосунку для ведення електронної медичної документації, який орієнтований на потреби медичних закладів малого масштабу, приватних клінік або індивідуальної лікарської практики.

У процесі реалізації було послідовно опрацьовано всі ключові етапи життєвого циклу програмного забезпечення.

У першому розділі виконано аналіз предметної області. Виявлено основні функціональні потреби користувачів — адміністраторів і лікарів, описано ролі та сценарії їх взаємодії із системою. Побудовано UML-діаграми [1] (прецедентів, діяльності, послідовності), які допомогли формалізувати логіку роботи системи та уточнити її функціональні вимоги.

У другому розділі створено концептуальну модель даних, яка охоплює основні сутності: пацієнтів, лікарів, діагнози, рецепти, медичні записи. На її основі реалізовано фізичну структуру бази даних у середовищі MySQL, забезпечено налаштування зовнішніх ключів, обмежень цілісності та структуру взаємозв’язків між таблицями. Окрема увага приділена коректному оформленню зв’язків багато-до-одного, наприклад, між записом і пацієнтом, діагнозом, рецептом.

У третьому розділі реалізовано програмну частину системи. Архітектура побудована за класичною трирівневою клієнт-серверною моделлю: інтерфейс користувача (на базі Windows Forms), рівень логіки доступу до даних (AccessComponent) і модельний рівень. Розроблено набір форм для введення та перегляду даних: реєстрація пацієнтів, створення медичних записів, додавання діагнозів і рецептів. Інкапсульована логіка взаємодії з базою забезпечує чисту архітектуру, повторне використання коду та зручне тестування.

У четвертому розділі проведено комплексне тестування системи, яке охоплювало:

- перевірку функціональних сценаріїв (авторизація, додавання, редагування даних);
- валідацію введення (обов’язкові поля, формат дати, коректність ID);
- обробку помилок (винятки при підключенні до БД, неправильні запити);
- оцінку інтерфейсу з точки зору зручності для медичного персоналу;
- базову перевірку продуктивності при роботі з великим обсягом записів.

За результатами тестування підтверджено стабільну роботу всіх основних модулів, правильну обробку даних і зручність використання для кінцевого користувача. Також було визначено вимоги до апаратного й програмного забезпечення, розроблено діаграму розгортання, а також сформовано інсталяційний пакет системи.

Система «Medix» успішно пройшла всі етапи проєктування, реалізації та тестування. Вона забезпечує повноцінне функціональне середовище для роботи з електронною медичною документацією, поєднуючи зручний інтерфейс, ефективну взаємодію з базою даних і дотримання принципів модульності.

Розроблене рішення є технічно завершеним, має просту структуру розгортання, підтримує масштабування й може бути адаптоване до потреб більш складних клінічних систем.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Уніфікована мова моделювання (Unified Modeling Language - UML) [Електронний ресурс] – Режим доступу: https://www.maxzosim.com/unifikovana-mova-modeluvannia/

2. Основи UML [Електронний ресурс] – Режим доступу: https://docs.kde.org/trunk5/uk/umbrello/umbrello/uml-elements.html

3. Посібник «Проектування інформаційних систем» / Макаренко [Електронний ресурс] – Режим доступу: https://elearning.sumdu.edu.ua/free_content/lectured:de1c9452f2a161439391120eef364dd8ce4d8e5e/20160217112601/content-20160217112601.pdf

4. Інтерфейс користувача [Електронний ресурс] – Режим доступу: https://ube.nlu.org.ua/article/%D0%86%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81%20%D0%BA%D0%BE%D1%80%D0%B8%D1%81%D1%82%D1%83%D0%B2%D0%B0%D1%87%D0%B0

5. Visual Studio IDE documentation [Електронний ресурс] – Режим доступу: https://learn.microsoft.com/uk-ua/visualstudio/ide/?view=vs-2022

6. Microsoft Docs: Windows Forms documentation [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/en-us/dotnet/desktop/winforms/

7. MySQL 8.0 Reference Manual [Електронний ресурс] – Режим доступу: https://dev.mysql.com/doc/

8. Нейгел К., Ивьен Б., Глинн Д., Уотсон К., Скиннер М. **C# 5.0 и платформа .NET 4.5 для профессионалов**. – М.: Вильямс, 2014. – 1440 с.

9. Хейлперс Т. **C# 9 and .NET 5 - Modern Cross-Platform Development**. – Birmingham: Packt Publishing, 2020. – 822 с.

10. Sharp J. **Microsoft Visual C# Step by Step**. – 10th Edition – Microsoft Press, 2022. – 832 с.

11. Entity Framework Documentation [Електронний ресурс] – Режим доступу: https://learn.microsoft.com/en-us/ef/

12. Безпека медичних інформаційних систем [Електронний ресурс] – Режим доступу: https://medstandards.com.ua/category/information-security/

13. C# Data Access (ADO.NET) [Електронний ресурс] – Режим доступу: https://learn.microsoft.com/en-us/dotnet/framework/data/adonet/

ДОДАТОК А

Sql скрипт для створення бази даних та її таблиць, а також внесення умовно-постійної інформації

Сторінок - 3

Київ – 2025


```
CREATE DATABASE IF NOT EXISTS medical_clinic;
USE medical_clinic;

-- Таблиця: administrator
CREATE TABLE administrator (
    AdminID VARCHAR(50) PRIMARY KEY,
    LastName VARCHAR(100),
    FirstName VARCHAR(100),
    MiddleName VARCHAR(100),
    Position VARCHAR(100),
    PhoneNumber VARCHAR(20)
);

-- Таблиця: doctor
CREATE TABLE doctor (
    DoctorID VARCHAR(50) PRIMARY KEY,
    LastName VARCHAR(100),
    FirstName VARCHAR(100),
    MiddleName VARCHAR(100),
    Position VARCHAR(100),
    PhoneNumber VARCHAR(20),
    Specialization VARCHAR(100)
);

-- Таблиця: patient
CREATE TABLE patient (
    PatientID VARCHAR(50) PRIMARY KEY,
    LastName VARCHAR(100),
    FirstName VARCHAR(100),
    MiddleName VARCHAR(100),
    MedicalCardNumber VARCHAR(50),
    Address VARCHAR(255),
    Email VARCHAR(100)
);

-- Таблиця: diagnosis
CREATE TABLE diagnosis (
    DiagnosisID VARCHAR(50) PRIMARY KEY,
    Name VARCHAR(255),
    Description TEXT
);

-- Таблиця: prescription
```

```
CREATE TABLE prescription (
    PrescriptionID VARCHAR(50) PRIMARY KEY,
    RecordID VARCHAR(50),
```



```
    PrescriptionDate DATE,
    Medication TEXT,
    Dosage VARCHAR(100),
    Instructions TEXT
);

-- Таблиця: medicalrecord
CREATE TABLE medicalrecord (
    RecordID VARCHAR(50) PRIMARY KEY,
    OpeningDate DATE,
    Notes TEXT,
    DiagnosisConclusion TEXT,
    PrescriptionID VARCHAR(50),
    PatientID VARCHAR(50),
    DiagnosisID VARCHAR(50),
    FOREIGN KEY (PrescriptionID) REFERENCES prescription(PrescriptionID),
    FOREIGN KEY (PatientID) REFERENCES patient(PatientID),
    FOREIGN KEY (DiagnosisID) REFERENCES diagnosis(DiagnosisID)
);

-- Таблиця: patientlog
CREATE TABLE patientlog (
    LogID INT AUTO_INCREMENT PRIMARY KEY,
    PatientID VARCHAR(50),
    ActionType VARCHAR(255),
    ActionTime TIMESTAMP,
    FOREIGN KEY (PatientID) REFERENCES patient(PatientID)
);
```

Додавання інформації в медичні картки

```
USE medical_clinic;

INSERT INTO medicalrecord (
    RecordID,
    OpeningDate,
    Notes,
    DiagnosisConclusion,
    PrescriptionID,
    PatientID,
    DiagnosisID
) VALUES
(
    1,
```

```
        '2025-05-18',
        'Пацієнт скаржиться на головний біль протягом тижня. Біль пульсуючого характеру, супроводжується нудотою.',
        'Гіпертонічний криз. Рекомендовано контроль тиску, зміна способу життя.',
        'PR001',
```



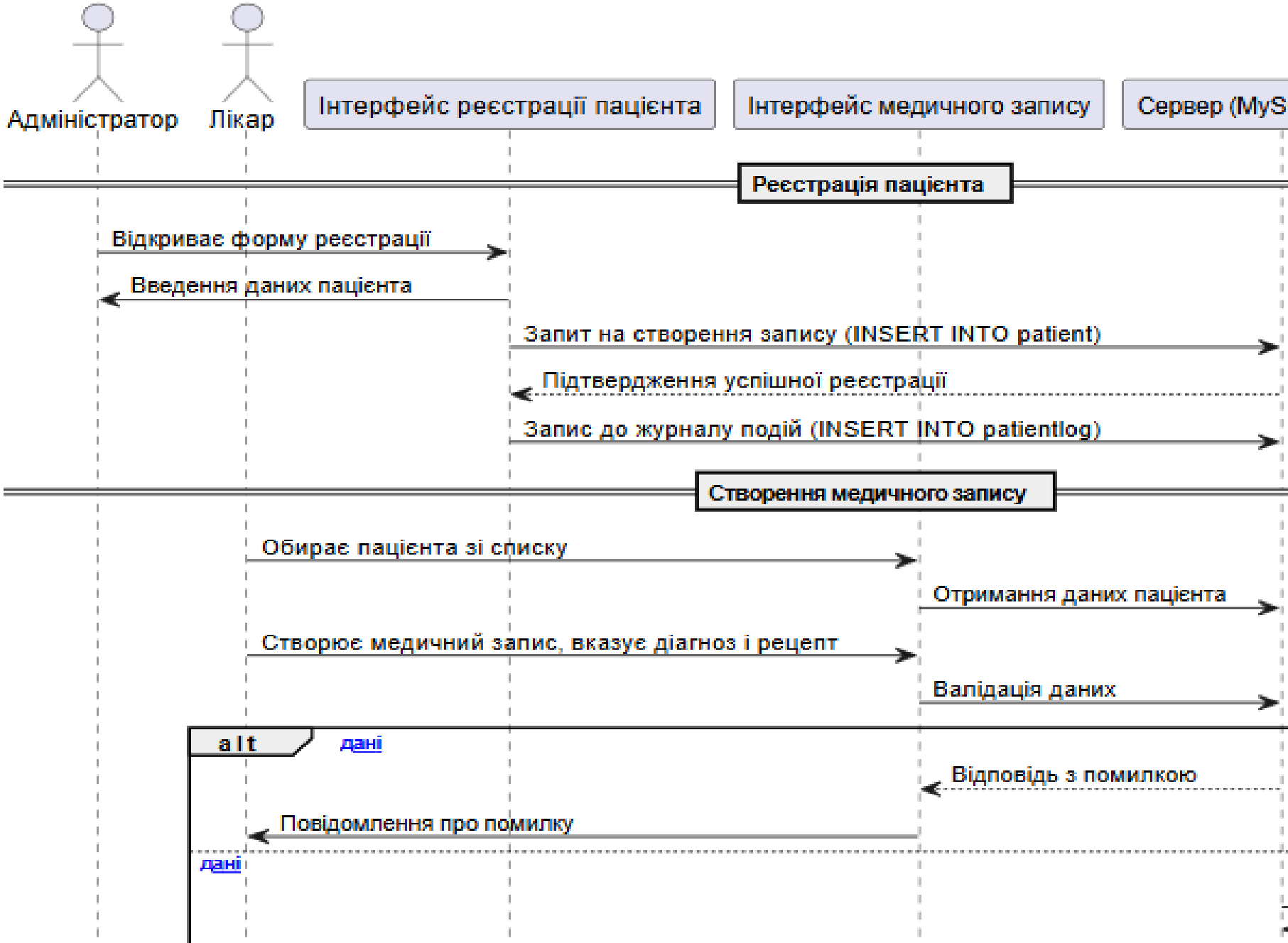
```
        'PAT001',
        'DIA001'
    ),
    (
        2,
        '2025-05-17',
        'Пацієнтка відзначає біль у горлі, температуру до 38.5°C, кашель та слабкість.',
        'Гострий фарингіт. Призначено антибіотикотерапію та полоскання горла.',
        'PR002',
        'PAT002',
        'DIA002'
    ),
    (
        3,
        '2025-05-16',
        'Пацієнт скаржиться на біль у попереку після фізичного навантаження. Біль посилюється при русі.',
        'М’язово-скелетне перенавантаження. Рекомендовано фізіотерапію та НПЗЗ.',
        'PR003',
        'PAT003',
        'DIA003'

    );
```

ДОДАТОК Б

Діаграма послідовності (Sequence Diagram)

Сторінок - 1

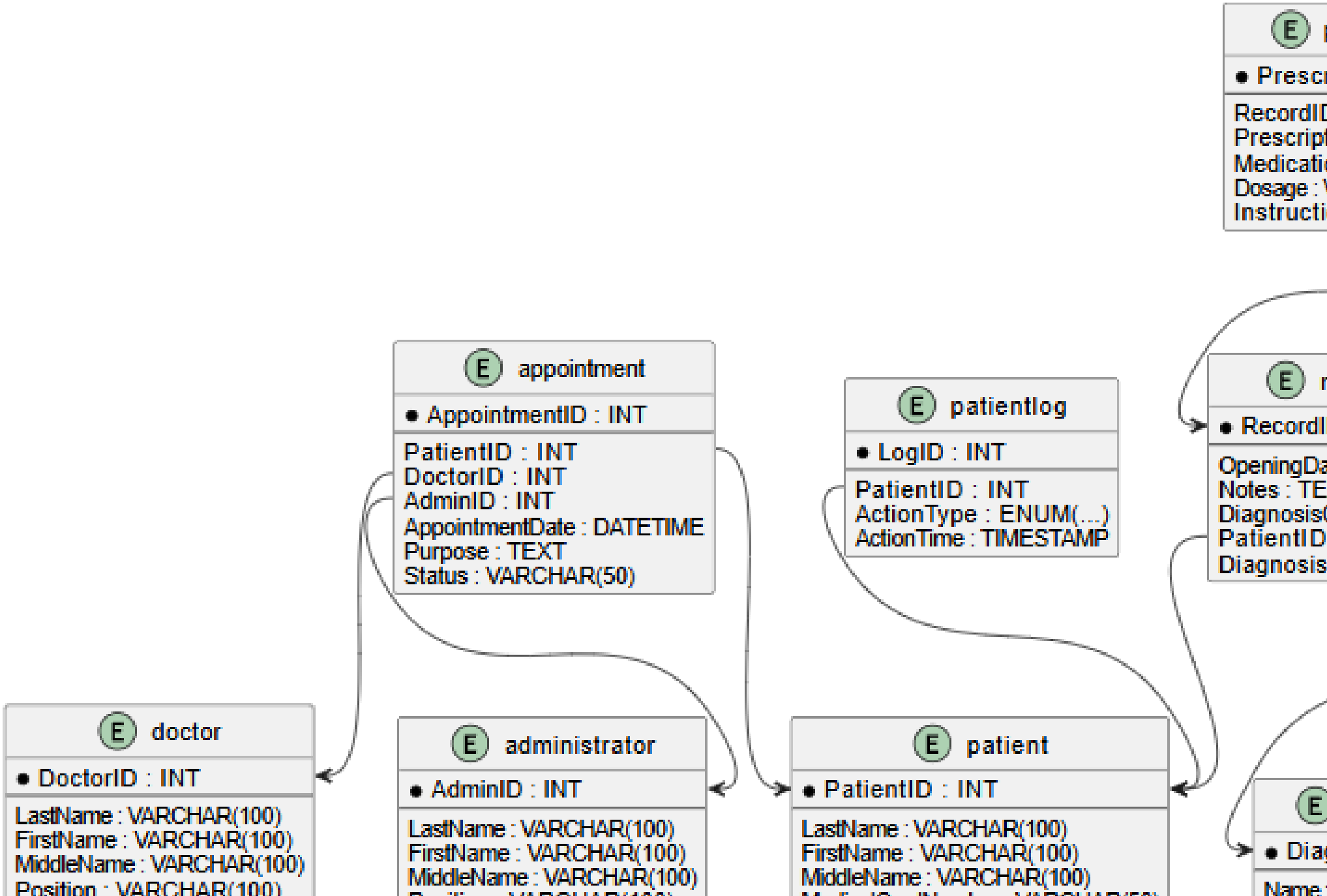
Київ – 2025


Адміністратор
Лікар
Інтерфейс реєстрації пацієнта
Інтерфейс медичного запису
Сервер (MyS
Реєстрація пацієнта
Відкриває форму реєстрації
Введення даних пацієнта
Запит на створення запису (INSERT INTO patient)
Підтвердження успішної реєстрації
Запис до журналу подій (INSERT INTO patientlog)
Створення медичного запису
Обирає пацієнта зі списку
Отримання даних пацієнта
Створює медичний запис, вказує діагноз і рецепт
Валідація даних
alt
дані
Відповідь з помилкою
Повідомлення про помилку
дані

ДОДАТОК В

ER-модель інформаційної бази

Київ – 2025


E
• Prescr
RecordID
Prescript
Medicati
Dosage : V
Instructi
E appointment
• AppointmentID : INT
PatientID : INT
DoctorID : INT
AdminID : INT
AppointmentDate : DATETIME
Purpose : TEXT
Status : VARCHAR(50)
E patientlog
• LogID : INT
PatientID : INT
ActionType : ENUM(...)
ActionTime : TIMESTAMP
E r
• RecordI
OpeningDa
Notes : TE
DiagnosisC
PatientID
Diagnosis
E doctor
• DoctorID : INT
LastName : VARCHAR(100)
FirstName : VARCHAR(100)
MiddleName : VARCHAR(100)
Position : VARCHAR(100)
E administrator
• AdminID : INT
LastName : VARCHAR(100)
FirstName : VARCHAR(100)
MiddleName : VARCHAR(100)
E patient
• PatientID : INT
LastName : VARCHAR(100)
FirstName : VARCHAR(100)
MiddleName : VARCHAR(100)
E
• Diag
Name :