Київський національний університет імені Тараса Шевченка

Факультет комп'ютерних наук та кібернетики

Кафедра математичної інформатики

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

за освітньо-професійною програмою «Інформатика»
спеціальності 122 «Комп'ютерні науки»
на тему:

Розробка єдиного програмно-алгоритмічного середовища візуалізації та комп'ютерного моделювання для створення систем оздоровлення військовослужбовців

Студента 4-го курсу
Богусевича Олексія Олександровича

_____
(підпис)

Науковий керівник:
Терещенко Василь Миколайович

_____
(підпис)

Робота заслухана на засіданні кафедри математичної інформатики та рекомендована до захисту в ЕК, протокол № …… від ..... 2021р.

Завідувач кафедри Терещенко В.М.

Київ – 2021

РЕФЕРАТ

Робота складається зі вступу, 4 розділів, висновків, списку використаних джерел (10 найменувань). Робота містить 11 рисунків, 3 таблиці. Загальний обсяг становить 45 сторінок, основний текст роботи викладено на 33 сторінках.

Ключові слова: АРХІТЕКТУРНИЙ ДИЗАЙН, БАЗИ ДАНИХ, ВЕБ-РОЗРОБКА, ЗБІР ІНФОРМАЦІЇ, ДІАГНОСТИКА, МЕДИЧНА СТАТИСТИКА, ОБ'ЄКТНО-РЕЛЯЦІЙНЕ ВІДОБРАЖЕННЯ, ОБРОБКА ВІДЕОПОТОКУ, ТЕСТУВАННЯ, ХМАРНІ ПЛАТФОРМИ

Об'єктом роботи є процес розв'язування задачі введення в практичне застосування системи реабілітації для військовослужбовців, що дозволяє автоматично замірювати кути повороту та нахилу голови для визначення стану окремих відділів опорно-рухового апарату досліджуваного пацієнта. Предметом роботи є проектування схеми бази даних для зберігання інформації про суб'єктів системи, розробка рольової моделі й реалізація механізмів аутентифікації та авторизації на її основі, створення зручного інтерфейсу користувача для управління системою, забезпечення інтеграції між різними компонентами системи, імплементація підсистеми оповіщень користувачів, а також впровадження інструментарію для швидкого розгортання системи та постачання на цільові кінцеві пристрої.

Метою роботи є аналіз існуючої системи штучного інтелекту та проектування, розробка і реалізація засобів для її використання персоналом медичних закладів.

Методи розроблення: розробка веб-додатків, розробка хмарних додатків, проектування баз даних.

Інструменти розроблення: безкоштовне, вільно поширюване інтегроване середовище розробки Visual Studio Code, розширена версія Enterprise для інтегрованого середовища розробки Visual Studio, система управління базами даних Microsoft SQL Server Management Studio, портал хмарної платформи

Microsoft Azure, інтерфейс командного рядку Azure CLI для управління ресурсами хмарної платформи Microsoft Azure, вільне інтегроване середовище розробки графічних інтерфейсів Qt Designer, утиліта Docker Desktop для створення та поширення контейнеризованих додатків.

Взаємозв’язок з іншими роботами: за основу взято систему штучного інтелекту, створену кандидатом фізико-математичних наук Коцуром Д. В. з використанням набору алгоритмів, розроблених професором Терещенко В. М. [1].

Результати роботи: спроектовано базу даних для збереження інформації про користувачів, пацієнтів, лікарів, дослідження, відділення та допоміжних компонентів, розроблено універсальну, легко розширювальну систему для управління пацієнтами, лікарями, проведення досліджень, генерації звітів та складання інтерактивних планів лікування, забезпечено інструментарій для швидкого розгортання системи, проведено модульне, інтеграційне та валідаційне тестування.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

БД — база даних

ОРС – опорно-рухова система

СУБД — система управління базами даних

API – Application Programming Interface

UI – User Interface

UML – Unified Modeling Language

UX – User Experience

ВСТУП

Оцінка сучасного стану об’єкта розробки. З кожним роком технологічний прогрес набуває все більшого впливу на сфери людського існування. На перший план виходить автоматизація ключових процесів, одним із яких є надання медичних послуг, де роль людського фактору важко недооцінити, адже найменша затримка або похибка може коштувати людині життя. Мінімізація впливу медичного експерта на діагностику та результат лікування є однією з першочергових задач, що постають в сфері медичних інформаційних технологій. До таких задач також відноситься організація діловодства лікарів, ведення клінічного обліку та обробка медичної статистики. Вирішення цих задач надає можливість проведення специфічних лабораторних досліджень, підвищує точність та ефективність надання послуг, збільшує доступність медичних експертів і допомагає в організації обслуговування пацієнтів. Важко собі уявити, якою була б зазначена сфера за умови відсутності таких інструментів як рентгенографія, електрокардіографія або ж електроенцефалографія, за умови відсутності аналітичних та прогностичних алгоритмів машинного навчання, за умови відсутності розподілених сховищ даних для зберігання медичної інформації.

Актуальність роботи та підстави для її виконання. Діагностика опорно-рухового апарату є ключовим етапом у лікуванні захворювань хребта, до яких відносяться сколіоз, кіфоз і протрузія, та захворювань суглобів, до яких відносяться артрит, артроз та остеохондроз. Неправильно поставлений діагноз може звести нанівець усі спрямовані на одужання кооперативні зусилля лікаря та пацієнта, що в результаті призводить до неефективного використання часу працівника медичного закладу, недоцільних витратат грошових ресурсів з боку пацієнта, а в окремих випадках навіть може дати ускладнення і погіршити стан хворого.

Запропоновані методи машинного навчання та алгоритми заміру кутів нахилу й повороту, які лежать в основі інструменту діагностики розробленої

системи реабілітації, забезпечують високу точність під час обстежень відповідних відділів опорно-рухового апарату, що в кінцевому результаті дозволяє запобігти постановці неправильного діагнозу та не допустити описаних вище можливих негативних наслідків як з боку пацієнта, так і з боку лікаря. Зазначений інструмент інтегровано з іншими компонентами системи, серед яких є портал для управління суб'єктами системи та підсистема сповіщень, що вкупі визначає повнофункціональний відкритий для розширення продукт, готовий до використання у спеціалізованих медичних закладах.

Мета й завдання роботи. Метою роботи є проектування схеми бази даних для розподіленого зберігання медичної інформації, реалізація функціоналу управління суб'єктами системи за допомогою уніфікованого єдиного інтерфейсу, розроблення рольової моделі та системи аутентифікації й авторизації для розмежування прав доступу, створення інструментів візуалізації динаміки змін на основі історії захворювань окремо взятого пацієнта, імплементація генерування звітних документів для формального оформлення діагностики, надання функціоналу побудови інтерактивного плану лікування пацієнтів, а також забезпечення цілісності та інтегрованості усіх компонентів системи.

Виконання роботи можна умовно поділити на такі етапи:

- Проектування БД для збереження даних про користувачів, пацієнтів, лікарів, відділень, обстежень та допоміжної інформації, необхідної для коректного функціонування системи;
- Реалізація засобів створення та первинного наповнення БД;
- Проектування рольової моделі та системи аутентифікації й авторизації;
- Розробка графічного інтерфейсу веб-додатку, шаблонів для поштової розсилки;

- Створення веб-додатку для управління суб’єктами системи, перегляду динаміки змін на основі історії захворювань, побудови інтерактивного плану лікування пацієнтів, генерування звітних документів для формального оформлення діагностики;
- Інтеграція з поштовим сервером для реалізації функціоналу поштової розсилки;
- Імплементація API-інтерфейсу для забезпечення інтегрованості інструменту діагностики з БД;
- Модифікація існуючого інструменту діагностики для взаємодії з API-інтерфейсом;
- Інтеграція з сервісами телефонії та створення системи SMS-повідомлень;
- Створення інструментарію для розгорнення системи та постачання на цільові кінцеві пристрої.

Зазначені етапи передбачають систематизацію існуючого досвіду створення прикладних програм та використання актуальних підходів у розробці подібних програмних засобів.

Об'єкт, методи й засоби розроблення. Для порталу управління суб’єктами системи зі сторони сервера було обрано мову C#, платформу .NET 5 та технологію ASP .NET Core, зі сторони клієнта використано мову програмування JavaScript, мови розмітки HTML5 та CSS3. Для забезпечення взаємодії між клієнтом та сервером обрано компонентно-орієнтований UI-фреймворк Blazor. За основу механізму аутентифікації було взято ASP .NET Core Identity на основі Cookies для веб-додатку та на основі JWT для API-інтерфейсу. Останній реалізовано з використанням мови C#, платформи .NET 5 та технології ASP .NET Core Web API. Для розробки інструменту діагностики використано мову Python та фреймворк Qt, дизайн додатку побудовано за допомогою утиліти Qt-designer. Для створення реляційної бази даних було обрано Code-first технологію Entity Framework Core 5, для

управління - СУБД Microsoft SQL Server. Інтеграція з сервісами поштової розсилки забезпечена за допомогою платформи SendinBlue, з сервісами телефонії – за допомогою платформи Twilio та безсерверного застосунку з типом Logic App від постачальника хмарних послуг Microsoft Azure. Для створення інструментів розгортання системи було використано Docker.

РОЗДІЛ 1. АНАЛІЗ ІСНУЮЧИХ РОЗРОБОК

1.1. Загальний огляд

В результаті проведеного аналізу конкурентів не було знайдено явних аналогів, які би покривали запропонований функціонал цілком та повністю. Більша частина існуючих на ринку рішень в першу чергу орієнтована на автоматизацію бізнес-процесів, що притаманні медичним установам, без акценту на конкретну предметну область у сфері, зокрема й на роботу з опорно-руховою системою. Найбільш розповсюдженими практиками та пропонованими послугами є впровадження інструментів для покриття таких потреб медичних установ як збереження даних про пацієнтів, їх реєстрація, запис до лікаря, управління штатом співробітників, ведення внутрішнього обліку, інтеграція з сервісами оплати рахунків тощо. Далі буде наведено перелік доступних програмних засобів, які в тій чи іншій мірі відповідають частковим вимогам, поставленим до описаної раніше системи.

1.2. MSK Solutions Pain Recovery Program

Musculoskeletal Solutions Pain Recovery Program – це рішення від Lifestyle Matrix, що пропонує набір пакетних засобів, у тому числі й програмних, що орієнтовані на всебічне відновлення пацієнта від проблем з опорно-руховою системою. Лікування позиціонується як усестороннє з огляду на комплексний підхід: розглядається «матриця відновлення», яка включає фізичний, поживний та психологічний підходи. Лікар отримує набір інструментів, що включає інструкцію з впровадження, брошуру з практичними рекомендаціями, ознайомчу інвентарну книгу з розкриттям матриці відновлення, презентаційну брошуру для пацієнта, електронний носій з навчальними відео, системний опитувальник, схему для побудови індивідуального плану лікування та програмний засіб для ознайомлення з теоретичним підґрунтям захворювань відповідного типу. Реалізація відбувається у такому порядку: пацієнт записується на прийом та в перше

відвідування заповнює опитувальник, що допомагає медичному експерту отримати необхідні дані про фізичний, психологічний та поживний стан хворого; в процесі розмови лікар застосовує терміни та знання, отримані в процесі вивчення змісту інвентарної книги, після чого надає пацієнтам матеріали для ознайомлення та заохочує вивчення теоретичних основ на основі рекомендацій з навчальних відео, що є важливим аспектом для розуміння важливості усіх складових комплексного підходу; після проведення вступних зустрічей кожний пацієнт отримує індивідуальний план на основі схеми.

Перевагою такого підходу у лікуванні є комплексність та усесторонняя націленість на результат, але система не пропонує діагностичних інструментів для фіксації більш точних фізичних характеристик, що в кінцевому результаті може стати причиною неефективно побудованого плану лікування. Окрім зазначеного, наявні програмні засоби набору не передбачають інструментів для ведення діловодства та перегляду медичної статистики.

1.3. EClinic Management System

EClinic Management System – програмний продукт від компанії Adroit Infosystems, який пропонує функціонал персоналізованої системи управління медичним закладом. Перелік доступних функцій рішення передбачає онлайн планування візитів до лікаря, підсистему внутрішнього листування, інтеграцію з модулем проведення лабораторних досліджень та сервісами страхування, інструменти для ведення внутрішнього обліку, побудову аналітичних звітів за здійсненними обстеженнями, засоби для візуалізації медичної статистики, підсистему телефонії та поштової розсилки, консультаційні послуги. Система надає віддалений доступ до даних пацієнта, відстежує ефективність розподілення фармацевтичних засобів і медичного обладнання, автоматизує управління фінансових модулів та допомагає в прийнятті рішень під час розробки комплексних політик медичних послуг.

Перевагою такого програмного засобу є великий спектр можливостей, але у той же самий час факторами не на користь рішення виступають досить висока ціна та складність у використанні.

1.4. Програмне забезпечення «Облік пацієнтів»

ПО «Облік пацієнтів» - це система для обліку пацієнтів в різноманітних медичних установах. Програмний засіб може створювати базу даних пацієнтів, зберігати в електронному вигляді повну картотеку пацієнтів, будувати список обстежень по кожному пацієнту, вести користувацькі словники (з інформацією про лікарів, відділення, організацій-партнерів тощо), допомагає автоматизувати роботу спеціалістів реєстратури та генерувати різноманітні документи – договори, результати обстежень та медичних заключень. За наявною інформацією система широко використовується у країнах Східної Європи з огляду на доступність та надійність.

Перевагами системи є відносно невисока ціна та широкий досвід практичного використання, але водночас рішення ґрунтується на застарілих технологіях, не містить інструментів діагностики і візуалізації медичної статистики та не передбачає можливості масштабування, адже розгортається локально на конкретному кінцевому користувацькому пристрої.

РОЗДІЛ 2. ОГЛЯД ФУНКЦІОНАЛУ СИСТЕМИ

2.1. Загальний огляд функціоналу системи

З точки зору користувача функціонально побудована система поділяється на дві основні компоненти: інструмент діагностики та портал. Перший дозволяє проводити обстеження окремих відділів опорно-рухового апарату людини, після чого аналіз відповідних результатів можна здійснити вже безпосередньо на порталі. Останній також надає можливості управління суб’єктами системи, візуалізації історій захворювань, генерування звітів, побудови інтерактивних планів лікування та виконання планів лікування пацієнтами. Для доступу до будь-якої із зазначених вище компонент спершу необхідно пройти процедури аутентифікації та авторизації.

2.2. Функціонал інструменту діагностики

Інструмент діагностики в початковому його вигляді було розроблено кандидатом фізико-математичних наук Коцуром Д. В. на основі алгоритмів, запропонованих професором Терещенко В. М. [1]. Описана далі технологія призначена для вимірювання та відстежування кутів нахилу голови у трьох площинах, що в медичних термінах визначає діагностику шийного відділу опорно-рухового апарату. В майбутньому планується розширити реалізований функціонал, так що для обстеження будуть доступні й інші відділи апарату. Наразі інструмент реєструє такі події (рис. 2.1):

- Відведення голови вправо та вліво – кут α (вісь рискання);
- Відведення голови вбік (праворуч, ліворуч) – кут β (вісь крену);
- Відведення голови вперед та назад – кут γ (вісь тангажу).

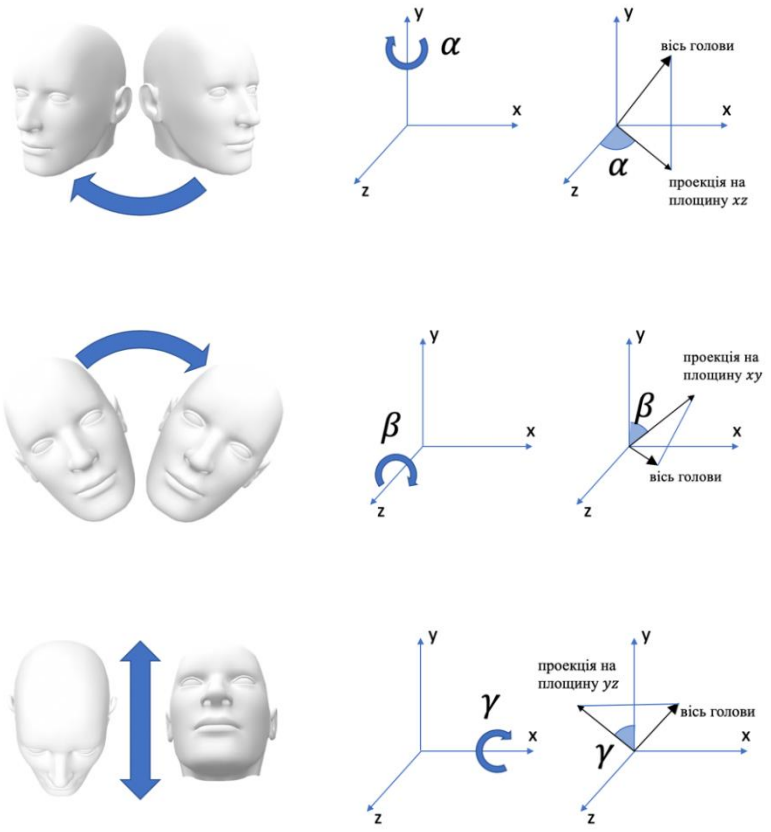


Рисунок 2.1 Кути нахилу голови

Виконуються заміри наступних величин: мінімальний, максимальний та середній кути повороту за кожною із осей (α, β, γ) у кожному напрямку (праворуч-ліворуч, вгору-вниз), а також кількість повторень поворотів.

Можливі типи вхідних даних:

- відео-потік із налаштованої веб-камери;
- попередньо записаний відеофайл (доступні формати: AVI, MP4).

Після вибору типу вхідних даних уповноважений проводити обстеження лікар отримує доступ до екрану моделювання, де має змогу в режимі онлайн спостерігати зміни кутів нахилу та повороту, а також їх динаміку та модель голови досліджуваного пацієнта (рис. 2.2).

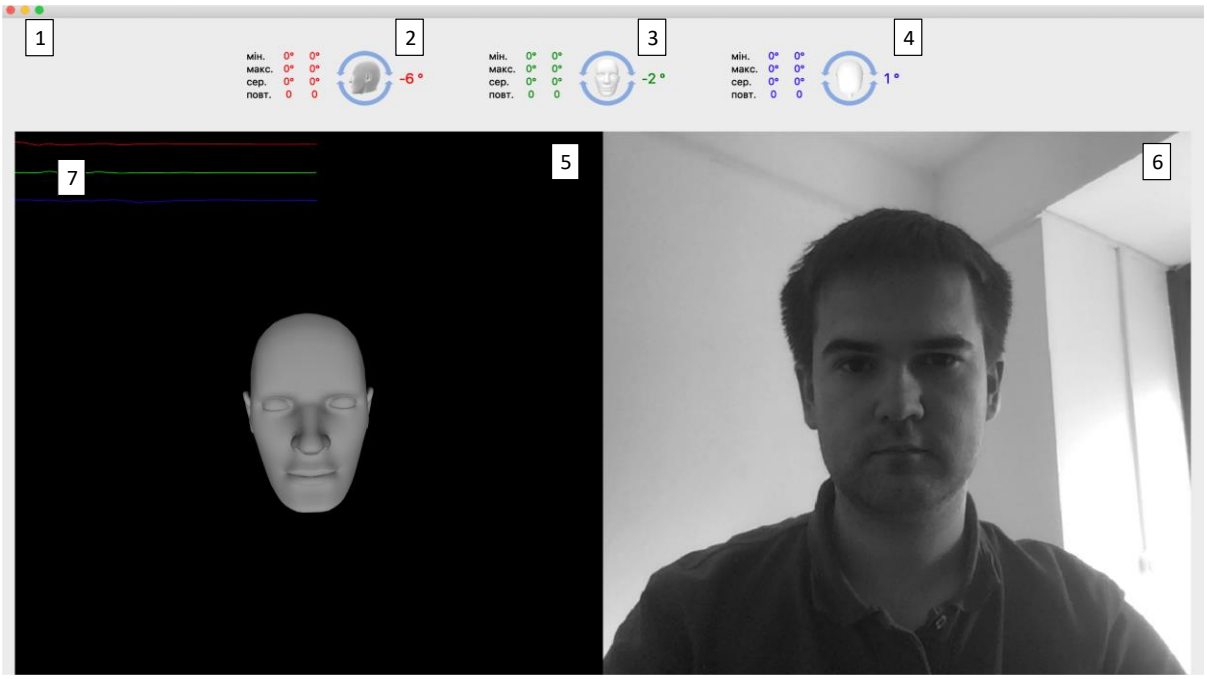


Рисунок 2.2 Елементи графічного інтерфейсу інструменту діагностики

Відповідність:

1. Панель відстеження динаміки поворотів;
2. Статистика нахилу голови вперед-назад;
3. Статистика нахилу голови в боки (праворуч та ліворуч);
4. Статистика поворотів голови вправо та вліво;
5. Вікно візуального моделювання голови людини «Аватар»;
6. Вікно виводу графічної інформації з камери (відео-потоку);
7. Графіки динаміки поворотів.

Існуючий інструмент діагностики для заміру кутів нахилу голови було доповнено системою аутентифікації та авторизації, а також вибору пацієнта зі списку пацієнтів, закріплених за відповідальним спеціалістом, що дозволило унеможливити несанкціонований доступ до технології, а також надало змогу встановити зв’язок між конкретним проведеним дослідженням та відповідним пацієнтом. Модель взаємодії з програмним засобом продемонстровано на нижченаведеній UML-діаграмі (рис. 1.3).

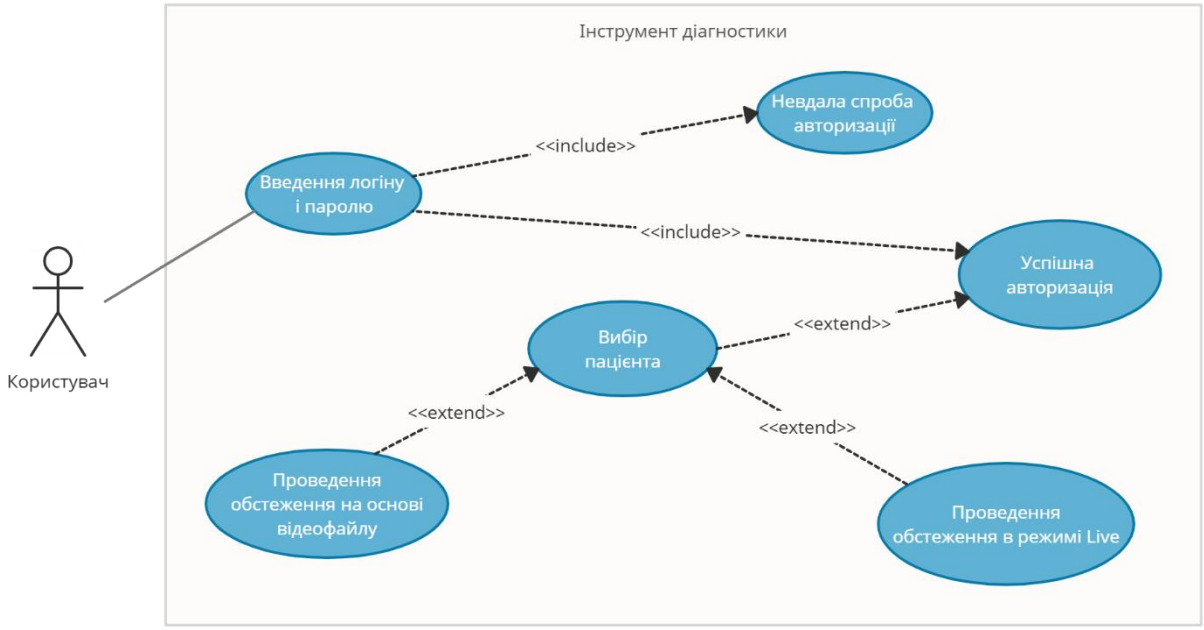


Рисунок 2.3 Use-Case діаграма для інструменту діагностики

2.3. Функціонал порталу системи

Повнота системи реабілітації забезпечується порталом для управління її суб'єктами. Розроблений функціонал передбачає його використання лікарями, пацієнтами та уповноваженими особами, що знаходяться на управлінських посадах медичних закладів. Можливості, які надає портал, визначаються приналежністю користувача до конкретної ролі в системі, що встановлюється в момент аутентифікації на стартовій сторінці веб-додатку (рис. 2.4), та цілком і повністю покривають задачі організації діловодства лікарів, обробки медичної статистики та взаємодії з пацієнтами.

Окрім вищенаведених медико-управлінських задач, що вирішуються з використанням веб-порталу системи, зазначений програмний засіб також передбачає наявність усіх необхідних інструментів з переліку стандартного користувацького набору, притаманного додаткам схожого профілю, а саме відновлення паролю за допомогою поштової адреси, редагування загальної

користувацької інформації профілю, зміну актуального паролю та поштової розсилки (рис. 2.5).

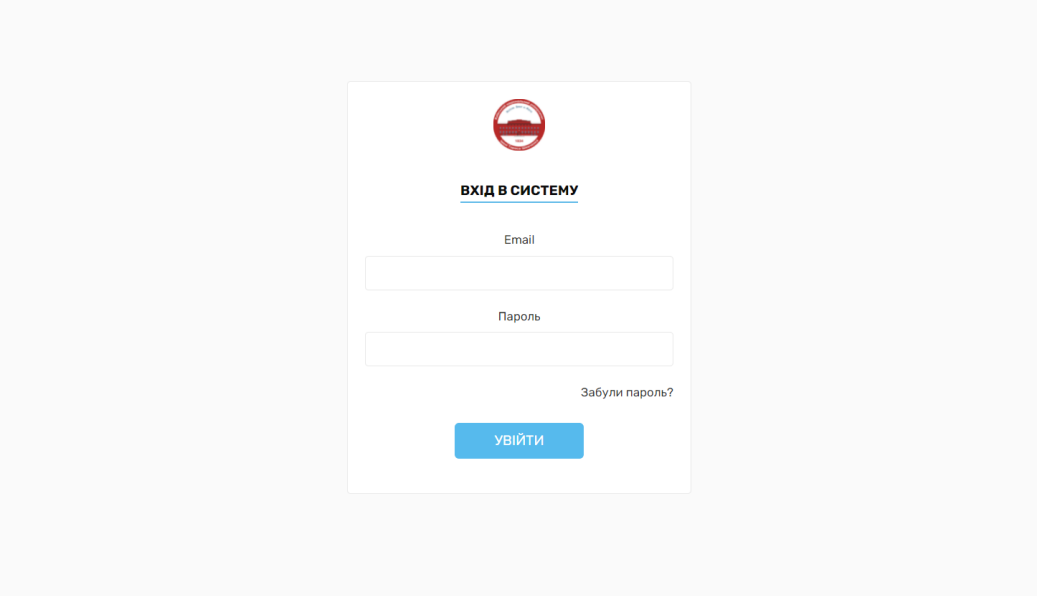


Рисунок 2.4 Сторінка входу в систему

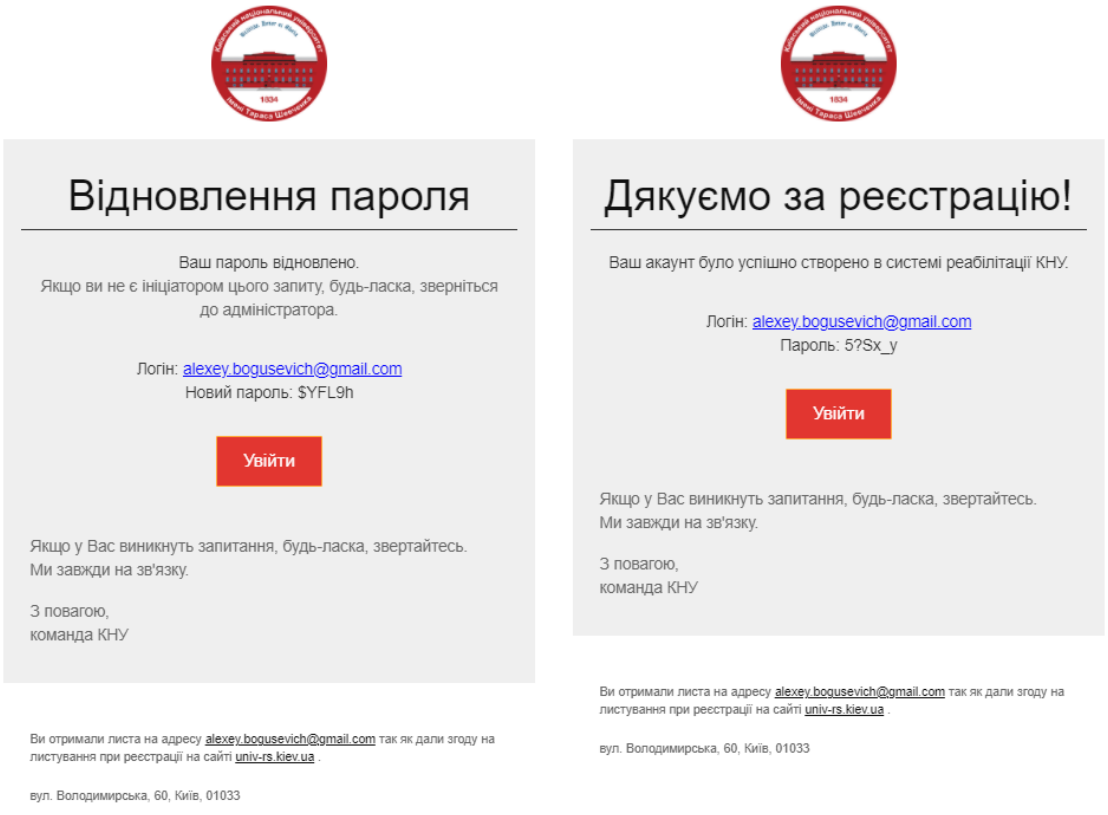


Рисунок 2.5 Приклади Email-повідомлень відновлення паролю та реєстрації

Після успішної аутентифікації користувач веб-порталу системи потрапляє на стартову сторінку, яка може різнитись в залежності від його ролі: адміністратори системи та медичні експерти в першу чергу отримують доступ до інформаційної панелі, де відображено загальні характеристики системи та список з останніх п’яти зареєстрованих пацієнтів (рис. 2.6), в той же час для самих пацієнтів доступною буде сторінка з призначеними ним вправами для виконання, що є частковим відображенням плану лікування, розробленим лікарями, які закріплені за цими конкретними пацієнтами (рис. 2.7).

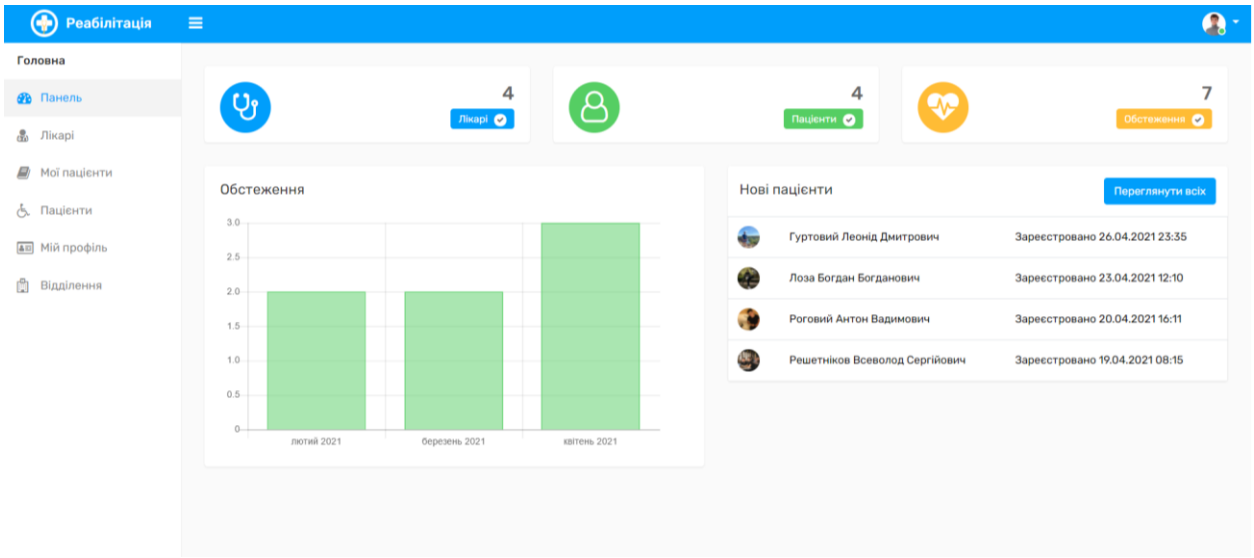


Рисунок 2.6 Інформаційна панель веб-порталу системи

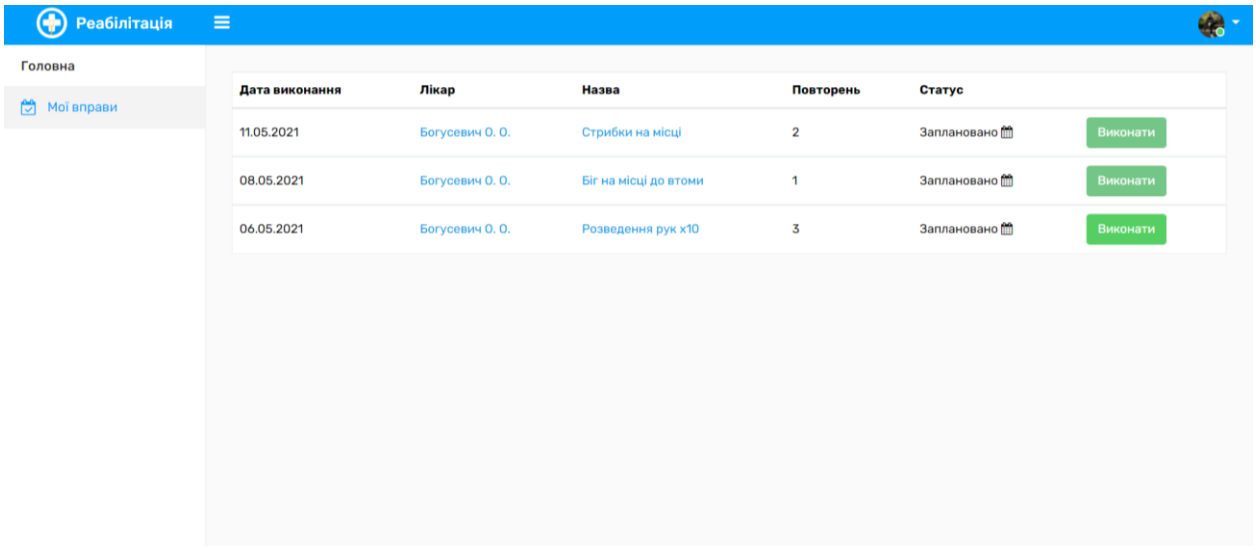


Рисунок 2.7 Сторінка веб-порталу системи «Мої вправи»

Сторінка «Мої вправи» визначає самостійну взаємодію пацієнта з системою реабілітації, що обмежується переглядом списку вправ для виконання з можливістю більш детально ознайомитись із кожною шляхом натискання на назву вправи, можливістю переходу на профіль медичного експерта, закріпленого за цим пацієнтом, який на його персональній сторінці призначив цю вправу для виконання, та можливістю присвоєння цій вправі статусу «Виконано». Відповідно завдання, яким не було присвоєно такий статус в означений день автоматично стають недоступними для пацієнта і помічаються як невиконані. Кожного дня о 9:00 пацієнти, яким було призначено вправи для виконання, отримують SMS-повідомлення з відповідним списком та побажанням гарного здоров’я.

На противагу, більш широким в контексті роботи з веб-порталом є спектр можливостей користувачів з роллю «Лікар». Медичні експерти отримують доступ до власного профілю та до профілю пацієнтів (рис. 2.8), які за ними закріплені, що включає в себе основну клінічну інформацію про людину (вага, зріст, сімейний стан, освіта, зайнятість, контакти, скарги, діагноз тощо), список проведених обстежень з можливістю переглянути деталі та згенерувати звіт по кожному (приклад звіту наведено в додатку А), візуальну інтерпретацію або ж прогрес в історії захворювання, що визначається графічним представленням динаміки змін впродовж проведених обстежень в розрізі типу, та списку призначених вправ для виконання з можливістю додавати, редагувати та помічати завдання як виконані на випадок проведення таких на місці огляду. Додатково лікар може редагувати інформацію про пацієнта в разі виникнення такої потреби, що до прикладу може статись внаслідок зміни ваги, сімейного стану або перегляду коректності поставленого діагнозу. В меню швидкого доступу, що стає видимим після натиснення на вертикальний символ в правому верхньому кутку інтерфейсу, також виключно для медичного експерта додається опція вибору завантаження інструменту діагностики, що не є можливим у випадку користувачів з іншими ролями. У профілі лікаря вказано загальні відомості (контактні дані, науковий ступінь,

професійні компетенції тощо), список усіх закріплених за ним пацієнтів та способи запису на прийом.

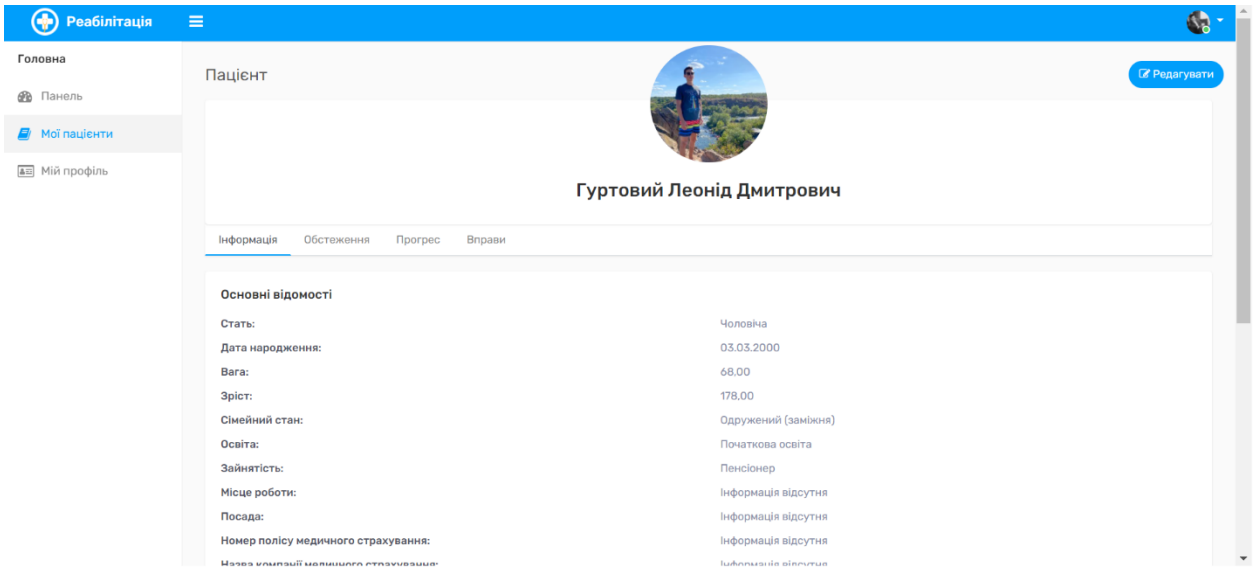


Рисунок 2.8 Сторінка веб-порталу «Профіль пацієнта»

Найбільш широкими повноваженнями наділено користувачів із роллю адміністратора системи: вони мають можливість переглядати список усіх пацієнтів, створювати нових, редагувати існуючих та видаляти помилково створених або неактуальних у разі виникнення такої необхідності. Окрім того, у таких користувачів є прерогатива перегляду списку усіх зареєстрованих в системі медичних експертів, редагування їх інформації та видалення у разі потреби. Також виключно адміністратори системи мають право призначати пацієнтів конкретним лікарям та надавати права адміністратора. Тільки в них є доступ до сторінки модерування клінічних відділень, де можна знайти інформацію про назву, місцезнаходження відділення, редагувати й створювати нові точки та видаляти існуючі.

Повну модель взаємодії з описаним компонентом системи продемонстровано на нижченаведеній UML-діаграмі (рис. 2.9).

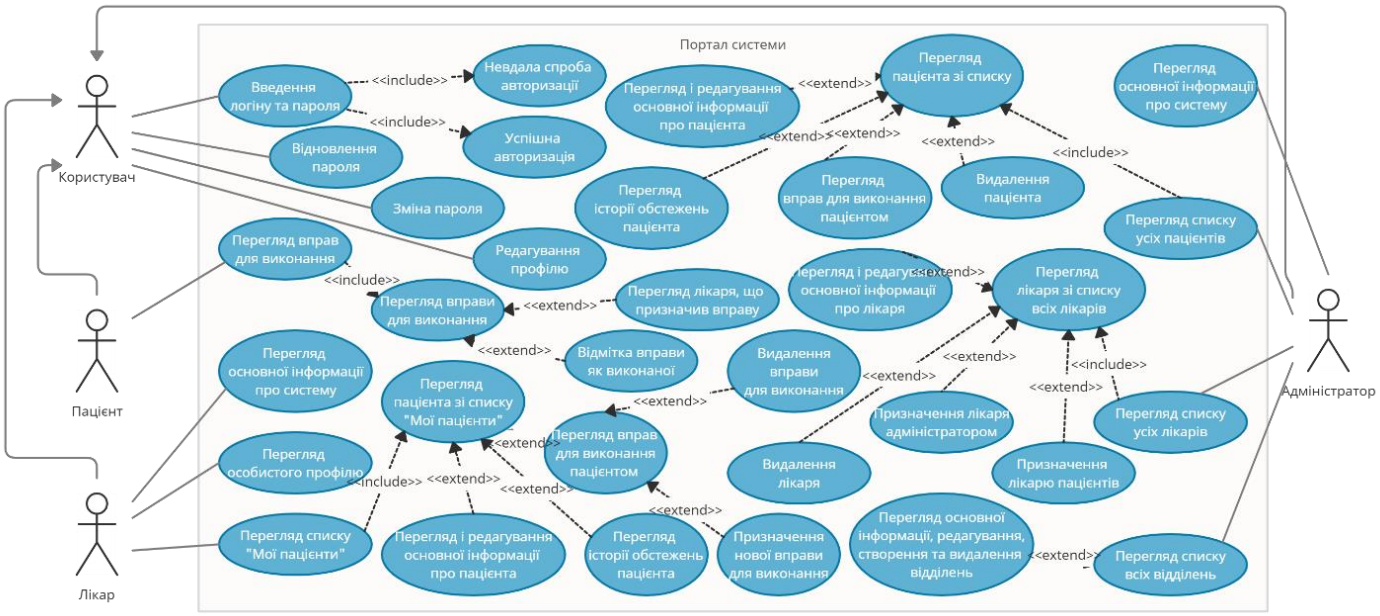


Рисунок 2.9 Use-Case діаграма для веб-порталу системи

РОЗДІЛ 3. ОГЛЯД АРХІТЕКТУРИ СИСТЕМИ

3.1. Загальний огляд архітектури системи

У більшості випадків розробка програмного забезпечення розпочинається із архітектурного дизайну. Цей процес визначає формат та об’єм робот, необхідних для побудови програмного продукту. Закладений із самого початку фундамент у вигляді схеми, що забезпечує оптимальну взаємодію між складовими системи, є ключовим фактором при побудові рішень будь-якої складності. На його основі будуються окремі фрагменти, що в загальній сукупності визначають кінцевий продукт. Одним із важливих концептуальних термінів в цьому контексті є рівень (англ. tier) – логічне розмежування компонент в застосунку або сервісі. В залежності від кількості рівнів вирізняють наступні види архітектури:

- **Однорівнева.** За умови такого підходу, логіка, графічний інтерфейс, БД та інші компоненти, що необхідні для повноцінної роботи програмного забезпечення, поміщується на один обчислювальний пристрій. Перевагами архітектурного шаблону є відсутність мережевих затримок та безпека даних на найвищому рівні. Водночас, бізнес втрачає контроль над екземплярами рішення, а швидкість роботи та коректність функціонування визначаються характеристиками кінцевого пристрою користувача.

- **Дворівнева** модель має відповідну кількість компонент та додає поняття клієнт-серверної моделі, що визначається як структура, яка розмежовує задачі та навантаження між постачальниками ресурсів або сервісів, тобто серверів, та запитувачів послуг, тобто клієнтів. Дворівнева архітектура передбачає розміщення графічного інтерфейсу та логіки на стороні клієнта, в той час як джерело інформації або сховище даних розташовано з боку сервера. Такий підхід зменшує кількість вразливостей, при цьому забезпечуючи мінімальне навантаження на мережу.

• **Трирівнева** схема фізично розділяє базу даних, графічний інтерфейс користувача та бізнес-логіку. Цей тип архітектури є досить популярним та широко використовуваним в індустрії: майже усі нескладні веб-сайти є частиною зазначеної категорії.

• **N-рівнева** архітектура містить чотири або більше компонент, якими в тому числі можуть бути кеш, черги повідомлень, балансувальники навантаження, пошукові сервери тощо. Усі популярні соціальні платформи (Instagram, Facebook, Twitter), індустріальні сервіси (Uber, Airbnb) та багатокористувацькі ігри (Pokémon Go) є представниками цього класу. Потреба в архітектурі такого виду пояснюється необхідністю дотримання принципів єдиного обов'язку та поділу зон відповідальності: кожна компонента має мати свою чітко визначену роль в системі. Відповідність таким вимогам робить рішення високо доступним та легко масштабованим.

Під час проектування архітектури системи реабілітації усі вищенаведені типи були розглянуті, їх переваги та недоліки враховані.

Результуюча модель складається з п'яти основних компонент, двох сторонніх сервісів та множини клієнтів, що визначено у переліку:

1. Реляційна SQL БД, розміщена на SQL сервері
2. Настільний додаток: інструмент діагностики опорно-рухового апарату
3. Веб-додаток для управління суб'єктами системи
4. API веб-інтерфейс для взаємодії інструменту діагностики з БД
5. Поштовий сервер SendinBlue
6. Azure Logic App з часовим тригером
7. Сервіс телефонії Twilio
8. Кінцеві користувацькі пристрої

Описана модель може бути представлена у вигляді діаграми. Номери на зображенні позначають відповідні компоненти (рис. 3.1).

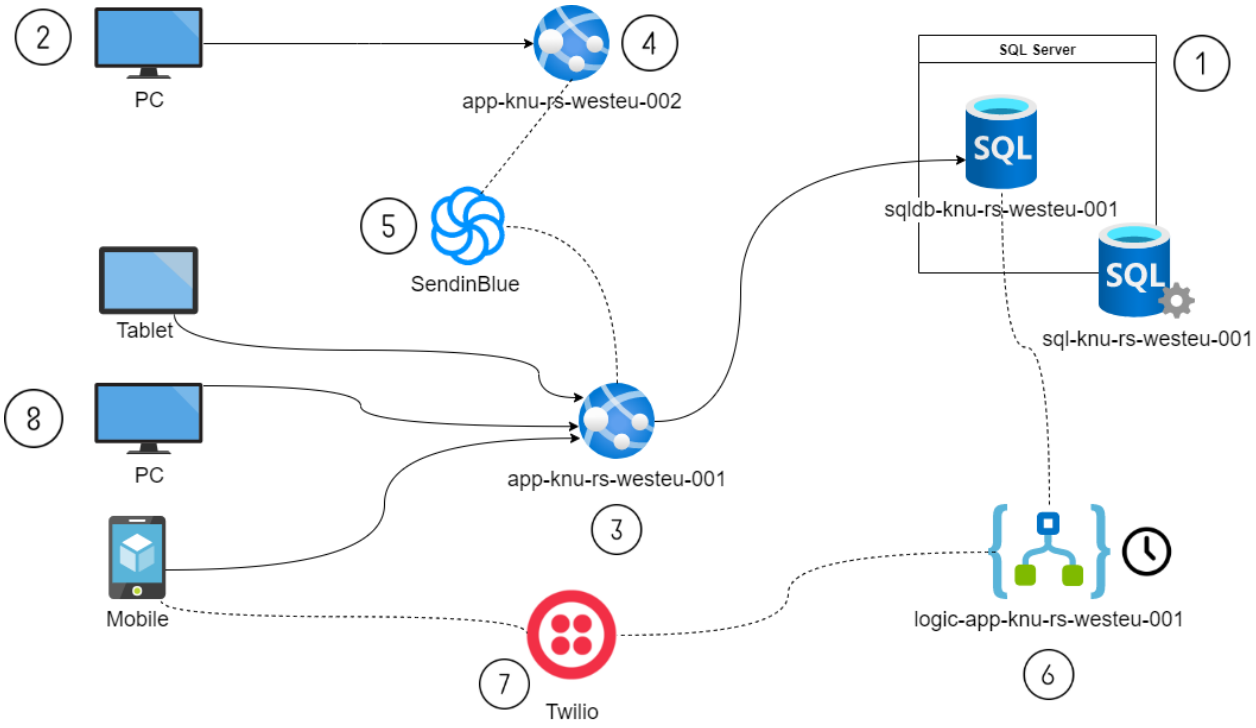


Рисунок 3.1 Схема архітектури системи

3.2. База даних

На перших етапах проектування одним із найбільш нагальних питань є вибір правильних технологій для зберігання внутрішньої інформації в системі. В першу чергу необхідно встановити загальний тип даних, що може бути визначений як структурований, напівструктурований (слабоструктурований) та неструктурований. Структуровані дані мають відповідати наперед визначеним правилам та конкретним схемам, не потребують попереднього опрацювання та зазвичай підлягають обробці SQL-подібними мовами. Неструктуровані гетерогенні дані є протилежністю структурованим, а напівструктуровані, прикладом яких є JSON та XML, представляють змішаний тип, особливості роботи з яким визначаються вимогами до системи. Після того, як встановлено тип даних, з якими планується робота в системі,

необхідно визначити тип сховища для зберігання цих даних. Сховища поділяють на реляційні та нереляційні (NoSQL). Перші використовуються в транзакційних системах із додатковими умовами забезпечення узгодженості та зберігання відношень між сутностями, другі – в навантажених системах з великою кількістю операцій зчитування та запису, де вимагається максимальна доступність компонент та легка масштабованість. З огляду на зазначені ключові характеристики того чи іншого типу сховища та особливостей медичної інформації було обрано модель зберігання структурованих даних в реляційному сховищі.

Під час розробки для створення, редагування та видалення даних, а також здійснення адміністративних процедур використовують спеціальні програмні засоби – системи управління базами даних (СУБД). Серед таких, категоризованих до роботи зі структурованими даними, можна виокремити декілька, що набули найбільшого розповсюдження [3]:

- **Oracle**: найпопулярніша комерційна СУБД для вирішення складних проблем систем середніх та великих розмірів, що доступна в багатьох конфігураціях, вимагає глибоких знань в предметній області та пропагує високу цінову політику;
- **MySQL**: простий у використанні недорогий програмний продукт з відкритим кодом, що набув широкого розповсюдження завдяки асоціативності з мовою програмування PHP, яка досить часто використовується у парі із зазначеною СУБД під час розробки веб-додатків;
- **SQL Server**: рішення від компанії Microsoft для роботи з великими об’ємами даних та побудови комерційних настільних додатків і веб-порталів, що використовується на різних рівнях у багатьох сферах;

Якщо до системи не поставлено вимог максимальної оптимізації в контексті роботи з базою даних, що до прикладу можна спостерігати в

програмних рішеннях фінансового сектору, то обробка даних шляхом виконання прямих SQL-запитів не є рекомендованою практикою, адже створює проблеми з розширенням системи та потребує додаткової логіки рівня застосунку задля забезпечення надійності виконуваних операцій. В більшості випадків, що не підпадають під описані критерії, використовується об’єктно-реляційне відображення, тобто технологія програмування, яка дозволяє створювати відповідність між реляціями реляційної бази даних й об’єктами об’єктно-орієнтованої мови та надає інтерфейс взаємодії з базою даних через її віртуальну об’єктну модель [10]. Однією з реалізацій таких відображень є технологія Entity Framework Core для платформи .NET. Вона є простою в розумінні та має широкий спектр можливостей для інтеграції. Саме зазначена технологія була обрана в якості прошарку між БД та іншими компонентами системи. Використання EF Core, що є технологією Microsoft, вплинуло на кінцеве рішення щодо СУБД: задля найкращої сумісності між технологіями одного постачальника вибір був зроблений на користь Microsoft SQL Server, який цілком та повністю відповідає вимогам системи.

В основі технології Entity Framework Core лежить поняття доменної моделі - множини бізнес-сутностей, які в тій чи іншій мірі покривають потреби користувачів програмного засобу та вирішують поставлені середовищем задачі. На етапі розробки наповнення зазначеної концептуальної множини може відбуватись за рахунок написання класів мовою програмування C# або ж автоматичного їх генерування на основі існуючої бази даних. У першому випадку використовується підхід Code First, де строго типізовані об’єкти рівня застосунку проектуються на таблиці реляційного сховища, в другому – Database First, що є зворотнім процесом до вищенаведеного. В обох варіантах синхронізація об’єктно-реляційного відображення відбувається в результаті проведення міграцій, що звіряють описову схему цільової бази даних з сутностями доменної моделі та застосовують відповідні зміні до структури таблиць сховища. Оперування зазначеними сутностями та, відповідно, програмна взаємодія з БД, що передбачає зчитування інформації, створення,

редагування й видалення рядків таблиць, можливе з використанням технології LINQ to SQL – підвиду інтегрованої в C# декларативної мови запитів LINQ до джерел інформації, що реалізують вбудований інтерфейс IEnumerable. З огляду на наявність перевірки відповідності типів часу компіляції, зрозумілий синтаксис та стандартизоване представлення, саме зазначеній технології було віддано перевагу на етапі архітектурного дизайну у контексті забезпечення взаємодії з базою даних. Оскільки розробка описуваної компоненти системи не передбачала використання існуючої схеми бази даних, в якості способу наповнення доменної моделі Entity Framework Core було обрано найбільш розповсюджений та прийнятий підхід Code First.

Спроектована схема мала в першу чергу передбачати можливість збереження інформації про пацієнтів, лікарів, обстеження та зв’язки між ними. Описані вимоги до системи були цілком та повністю задовільнені в процесі реалізації, додатково забезпечено представлення сутностям користувачів, ролей, кваліфікацій, оздоровчих вправ та відділень. Кінцевий перелік каталогів бази даних містить 18 таблиць, серед яких є автоматично згенеровані за допомогою шаблону ASP .NET Identity і програмно доповнені AspNetRoleClaims, AspNetRoles, AspNetUserClaims, AspNetUserLogins, AspNetUsers, AspNetUserTokens, та кастомні Clinics, DoctorPatients, Doctors, Educations, Patients, Qualifications, RecoveryDailyPlans, StudyDetails, StudyHeaders, StudySubtypes і StudyTypes. Структури центральних сутностей системи, якими є користувач, лікар та пацієнт, подано у відповідних таблицях 3.1, 3.2 та 3.3.

Назва поля	**Тип**	**Опис**
Id	uniqueidentifier	Ідентифікатор користувача
FirstName	nvarchar(MAX)	Ім’я
LastName	nvarchar(MAX)	Прізвище
MiddleName	nvarchar(MAX)	По-батькові
Address	nvarchar(MAX)	Адреса
Email	nvarchar(MAX)	Поштова адреса
PasswordHash	nvarchar(MAX)	Хеш пароля (PBKDF2)
PhoneNumber	nvarchar(MAX)	Номер телефону
Gender	int	Стать
Birthday	datetime2(7)	Дата народження
HasPhoto	bit	Наявність фотографії
PromotedToAdmin	bit	Приналежність до ролі «Адміністратор»

Таблиця 3.1 Атрибути сутності «Користувач»

Назва поля	**Тип**	**Опис**
Id	uniqueidentifier	Ідентифікатор лікаря
QualificationId	uniqueidentifier	Ідентифікатор посади
ClinicId	uniqueidentifier	Ідентифікатор відділення
UserId	uniqueidentifier	Ідентифікатор користувача
Biography	nvarchar(500)	Біографія
Competencies	nvarchar(500)	Професійні компетенції
Room	int	Номер аудиторії
Degree	nvarchar(100)	Науковий ступінь

Таблиця 3.2 Атрибути сутності «Лікар»

Назва поля	**Тип**	**Опис**
Id	uniqueidentifier	Ідентифікатор пацієнта
UserId	uniqueidentifier	Ідентифікатор користувача
Weight	decimal (18, 2)	Вага
Height	decimal (18, 2)	Зріст
Complaints	nvarchar(MAX)	Скарги
Diagnosis	nvarchar(MAX)	Діагноз
MaritalStatus	int	Сімейний стан
Occupation	int	Зайнятість
Job	nvarchar(MAX)	Місце роботи
Position	nvarchar(MAX)	Посада
EducationStatus	int	Освіта
Insuranse	nvarchar(MAX)	Поліс страхування
InsuranseCompany	nvarchar(MAX)	Назва страхової компанії
Passport	nvarchar(MAX)	Посвідчення особи
RegistrationDate	datetime2(7)	Дата реєстрації

Таблиця 3.3 Атрибути сутності «Пацієнт»

3.3. Інструмент діагностики

Попередньо розроблений інструмент діагностики є ключовим компонентом системи, що дозволяє моделювати рух частин та проводити обстеження відповідних відділів ОРС. Окрім застосування алгоритмічних розрахунків, програмний засіб також передбачає використання методів машинного навчання, що є лакмусовим папірцем для вибору Python в якості мови програмування під час фази системного дизайну з огляду на велику кількість загальнодоступних бібліотек з готовими інструментами для обробки статистичних даних та побудови моделей, одна із яких, 3DDFA, і була використана для розпізнавання обличь під варіативним кутом нахилу та побудови їх 3D-моделей за допомогою згорткової нейронної мережі. Алгоритмічно загальна схема рішення інструменту діагностики може бути розглянута як послідовність реалізації таких етапів:

1. Визначення місцезнаходження обличчя людини в кадрі за допомогою CNN-детектора бібліотеки OpenCV;
2. Застосування згорткової нейронної мережі для фіксації ключових точок [7];
3. Побудова тріангуляції для точок голови за допомогою МЄАС [2];
4. Реєстрація множини точок голови на основі проекції деформованої моделі у 3D та алгоритму «Ітеративна найближча точка» [8].
5. Застосування лінійного перетворення точок для визначення осі та орієнтації голови;
6. Проектування векторів, що відповідають осі та напрямку голови, для визначення кутів поворотів.

Оформлення графічного інтерфейсу для запропонованого інструменту у вигляді настільного додатку мовою Python може бути здійснено багатьма способами. Перелік найбільш розповсюджених фреймворків для розробки застосунків вищеописаного типу складається з таких варіантів:

- **PyQt**: багатоплатформний набір інструментів у відкритому доступі для побудови графічного інтерфейсу на основі віджетів; реалізовано як варіацію бібліотеки Qt від Nokia для мови програмування Python; вважається потужнішим за конкурентів завдяки наявності дизайнеру з можливостями кодогенерації;
- **Tkinter**: найбільш популярний пакет для розробки візуальної складової настільних додатків мовою Python; вирізняється простотою, обширною документацією та великою спільнотою, що можна пояснити відносно давньою датою створення;
- **PyGUI**: найпростіший багатоплатформний GUI-фреймворк, що був написаний виключно мовою Python, за рахунок чого забезпечується максимальна синхронізація між платформою та застосунком малими об’ємами коду.

Програмний засіб для проведення діагностики ОРС разом із надбудовою у вигляді аутентифікації не передбачає складної анімації та програмованої логіки переходів, а отже було вирішено, що використання дизайнеру для побудови форм із основними елементами інтерфейсу користувача не створить критичної різниці з аналогічним програмованим підходом, звідси як наслідок вибір було зроблено на користь бібліотеки PyQt.

Запис результатів обстеження, так само як і отримання даних для входу в систему та списку пацієнтів авторизованого медичного експерта, відбувається на основі взаємодії з відкритим API-інтерфейсом: після успішної аутентифікації подальші форми оперують ідентифікатором пацієнта та JWT-токеном, необхідним для доступу до API. Детальніше взаємодію з інтерфейсом описано у відповідному розділі.

3.4. Веб-портал системи реабілітації

Розробка центральної контрольної компоненти для взаємодії лікарів з пацієнтами та перегляду медичної статистики визначає основну частину роботи на даному етапі побудови системи. Серед поставлених вимог до програмного засобу виокремлюється незалежність від платформи та кінцевого пристрою користувача, що в значній мірі обумовлює вибір типу застосунку та виключає з множини можливих варіантів настільний та мобільний додатки. В обох випадках для відповідності вищенаведеним критеріям необхідно створювати програмний продукт для усіх найбільш розповсюджених операційних систем (відповідно Windows, Linux, MacOS та Android, iOS), що не є раціональним рішенням з точки зору процесу розробки, оскільки вимагає довготривалого залучення компетентних спеціалістів з кожної предметної області. Слід зазначити, що можливість побудови такого рішення у разі виникнення потреби не виключається у майбутньому. Наведені аргументи були враховані на етапі системного дизайну та у якості типу застосунку було обрано клієнт-серверний веб-додаток.

Кожен окремий веб-додаток потребує індивідуального підходу в плані вибору стеку технологій в залежності від вимог до функціоналу: обробка даних в режимі реального часу (прикладом якої є месенджери, багатокористувацькі ігри, колаборативні онлайн-редактори і стримінгові сервіси) зазвичай вимагає наявності стійкого з’єднання та неблокуючої технології, через що переважно реалізується з використанням Python, NodeJS, Spring Reactor, Play; в той же час як застосунки з активним навантаженням на оперативну пам’ять та процесор (прикладом яких є такі, що працюють з Big Data, виконують паралельні обчислення, реалізують моніторинг або аналітичні сервіси), де швидкість роботи та маніпуляція пам’яттю є критичними, в більшості випадків опираються на C++, Elasticsearch, Erlang, Go, Rust або Julia. Портал для управління суб’єктами системи реабілітації є прикладом звичайних CRUD-орієнтованих (створення, зчитування, редагування, видалення) додатків, а отже може бути реалізований за

допомогою таких технологій як Spring MVC, ASP.NET, Python Django, Ruby on Rails, PHP Laravel тощо. Для сумісності із наявною екосистемою Microsoft було віддано перевагу мові програмування C#, платформі .NET та технології ASP.NET Core разом із Server-side Blazor у якості прошарку між клієнтським кодом та бізнес-логікою сервера.

Оснований на загальномовній середі виконання CLR програмний фреймворк .NET, що був випущений в 2002 році компанією Microsoft, наразі користується популярністю серед широкого загалу завдяки швидкодії, потужності та різноманітності інструментарію для роботи як з веб-застосунками, так і з настільним програмним забезпеченням й хмарними сервісами. Платформа, що після низки варіацій (.NET Framework, .NET Core, .NET Standard) в 2020 році стала уніфікованою, наразі має потужну підтримку від Microsoft та кожного року оновлюється. Остання версія, .NET 5, і була використана під час розробки описуваної в розділі компоненти.

Стандартом створення динамічних веб-сервісів на платформі .NET є ASP.NET Core – надбудований багатоплатформний фреймворк з кодом у відкритому доступі для побудови сучасних готових до розгортання на хмарній платформі веб-застосунків, IoT рішень та мобільних додатків. Серед переваг зазначеної технології можна виокремити інтеграцію з багатьма популярними бібліотеками для написання клієнтського коду, вбудовану ін’єкцію залежностей (англ. Dependency Injection – шаблон проектування для досягнення слабко-зв’язного коду шляхом використання контейнера Inversion of Control), наявність модульного HTTP пайплайну та можливості самостійного хостингу у власному процесі. Програма на основі ASP.NET Core представляє собою консольний додаток, який створює веб-сервер в головному методі. Усі запити до сервера перенаправляються до застосунку та проходять через проміжний прошарок (middleware), де поетапно застосовується асинхронна логіка обробки, після чого вже у формі налаштованих екземплярів класу HttpContext потрапляють до методів з основною бізнес-логікою.

Разом із ASP.NET Core зі сторони сервера для створення інтерактивного графічного веб-інтерфейсу користувача зі сторони клієнта був використаний компонентно-орієнтований UI-фреймворк Blazor. Основна ідея подібних технологій (до яких також відносяться React, AngularJS, VueJS) полягає в повторному використанні окремих фрагментів візуальної складової застосунків (панелей, форм, таблиць, графіків тощо). Характерною ознакою саме цієї бібліотеки є можливість побудови елементів інтерфейсу без залучення мови програмування JavaScript, замість якої тут виступає C#, що привносить потужність платформи .NET у клієнтський код. Фреймворк має дві моделі хостингу: Blazor WebAssembly та Blazor Server. В першому випадку додаток повністю завантажується в пам’ять процесу веб-браузера та виконується на стороні клієнта, після чого запити до сервера надходять лише у разі необхідності взаємодії зі сторонніми сервісами. Такий підхід застосовується коли серед вимог до веб-додатку є забезпечення високої швидкості взаємодії із застосунком та слабка залежність від стану підключення до мережі. Недоліком є вивантаження серверного коду на кінцевий користувацький пристрій, що теоретично може призвести до проблем з безпекою за рахунок декомпіляції. Недоліки Blazor WebAssembly визначають переваги Blazor Server. Останній інтегрується зі стандартним ASP.NET Core, що дозволяє повторно використовувати бізнес-логіку сторони сервера в графічних компонентах Razor. Інтерактивний режим та двосторонній обмін даними тут забезпечується встановленням стійкого з’єднання на основі технології SignalR Core на початку користувацької сесії. SignalR Core – бібліотека для ASP.NET Core, що надає зручний програмний інтерфейс для клієнт-серверної взаємодії у режимі реального часу: в залежності від можливостей тієї чи іншої сторони технологія обирає оптимальний тип двосторонньої взаємодії з пріоритетного списку транспортів, який включає Long Polling, що періодично встановлює тривале одностороннє підключення для передачі нових даних у разі їх появи, Server Sent Events, що підписується на отримання оновлень через зарезервовані відкриті канали, та

WebSocket, що передбачає встановлення двонаправленого з’єднання, яким для обміну інформацією за потреби може користуватись як клієнт, так і сервер. У якості моделі для розробки веб-порталу системи було віддано перевагу Blazor Server в першу чергу через відсутність проблем з безпекою, дотримання якої є критичним у рішеннях медичної сфери.

Побудований застосунок відповідає шаблону Clean Architecture, яка передбачає, що архітектура повинна бути тестованою, не залежною від графічного інтерфейсу та баз даних, зовнішніх фреймворків й бібліотек. Такі вимоги виконуються завдяки розділенню на шари (рис. 3.2) й слідуванню правилу залежностей: внутрішні шари не повинні залежати від зовнішніх. Це правило дозволяє будувати системи, які легше підтримувати, оскільки модифікації в зовнішніх шарах не впливають на внутрішні. Таким чином до прикладу з’являється можливість змінювати інтерфейс без необхідності змінювати бізнес-логіку. Підхід був запропонований Робертом Мартіном [5] в 2011 році й в своєму оригінальному представленні виділяє чотири шари: Entities (спільна бізнес-логіка), Use Cases (логіка застосунку), Interface Adapters (зв’язуюча ланка між Use Cases й зовнішніми шарами) та Frameworks (графічний інтерфейс, робота зі сторонніми сервісами тощо), взаємодія між якими забезпечуються за рахунок інтерфейсів для вхідних запитів та вихідних відповідей на ці запити. Застосування шаблону під час побудови системи дозволило ефективно використовувати реалізовані сервіси як у контексті API-інтерфейсу, так і в процесі реалізації веб-додатку для управління суб’єктами системи, зробило можливим модульне ізольоване тестування окремих компонент та посприяло підвищенню швидкості реалізації функціоналу.

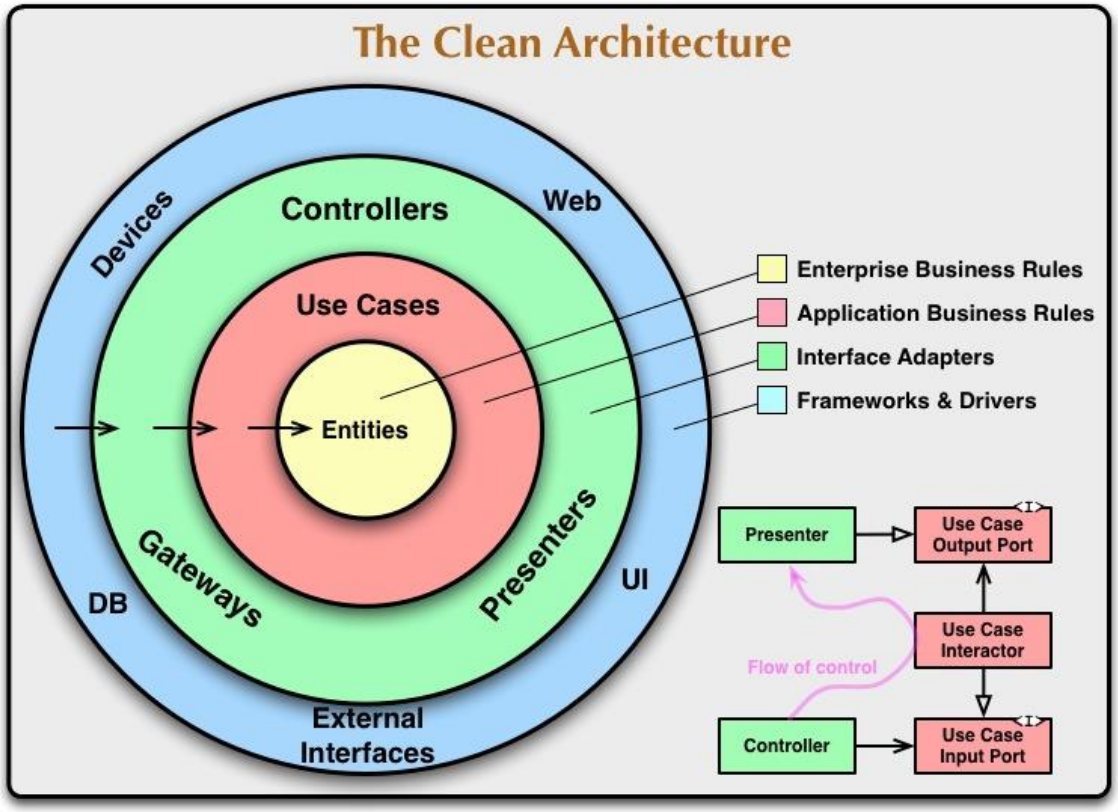


Рисунок 3.2 Схема моделі Clean Architecture

3.5. Web API

У відповідності до обраної N-рівневої архітектурної моделі, кожна компонента системи має мати єдиний обов’язок та визначену сферу відповідальності. Цей принцип пропагує слабку зв’язність між складовими (англ. loosely coupled), яка в кінцевому результаті дозволяє легко масштабувати рішення, робить його надійним та високодоступним. Такий підхід був застосований в процесі забезпечення взаємодії інструмента діагностики з базою даних для збереження інформації про обстеження пацієнтів: замість прямого доступу до сховища з настільного додатку було прийнято рішення розробити REST API, що дозволило абстрагуватись від конкретного типу БД та впровадити систему аутентифікації й авторизації на основі механізмів ASP.NET Core Identity.

REST API - це реалізація API-інтерфейсу, яка відповідає стандартам проектування REST, для яких характерними ознаками є гнучкість, відсутність стану в процесі комунікації клієнта та сервера, орієнтованість на протокол HTTP та асоціацію ресурсів з унікальним посиланням. Останній критерій звільнює сервер від знань про реалізацію клієнта, що в конкретному випадку дозволяє абстрагуватись від обстежень та інструментів для їх проведення. Розроблений інтерфейс надає авторизованим користувачам доступ до інформації про наявні типи обстежень (GET /studies/subtype), дозволяє отримати список закріплених за лікарем пацієнтів (GET /patients) та зберегти деталі проведеного дослідження (POST /studies). Авторизація реалізована відповідно до стандартного авторизаційного протоколу OAuth 2.0, за якого користувач надає свої дані для аутентифікації, якими в більшості випадків є логін та пароль, зовнішньому постачальнику індивідуальних особистостей, на що той відповідає в залежності від обраної моделі: у випадку Authorization Flow він повертає авторизаційний код, який клієнт може обміняти на токен доступу JWT до API; у випадку Implicit Flow клієнт отримує токен доступу напряму в результаті обробки першого ж запиту до авторизаційного серверу. JWT (JSON Web Token) – відкритий стандарт, що описує спосіб безпечної передачі даних мережею у вигляді об’єкту в форматі JSON. Такі об’єкти є верифікованими цифровим підписом, поставленим одним із способів: шифруванням секрету за допомогою HMAC або ж застосуванням алгоритму RSA / ECDSA для створення пари з публічного та приватного ключів. Вимоги до безпеки в системі цілком та повністю покриваються моделлю OAuth 2.0 Implicit Flow, тож під час реалізації з огляду на відсутність аргументів на користь ускладнення процесу авторизації перевагу було віддано саме цьому підходу з використанням алгоритму HMAC для створення цифрового підпису JWT.

3.6. Azure Logic App

Наявний функціонал призначення лікарями відновлювальних вправ для виконання пацієнтами був би неповним без автоматизованої підсистеми оповіщень. Оскільки жодна зі складових рішення не була прив’язана до кінцевих користувацьких пристроїв, реалізація зазначеної компоненти потребувала розробки самостійного програмного засобу із інтеграцією зі сторонніми комунікаційними сервісами. Для таких цілей було вирішено створити безсерверний хмарний застосунок.

Більшу частину ринку постачальників хмарних послуг поділено між трьома великими компаніями – Amazon (33%), Microsoft (20%) та Google (7%):

- AWS, створена в 2006 році компанією Amazon, є однією з перших хмарних платформ та наразі лідером у сфері саме завдяки своєму ранньому започаткуванню та довшій історії розвитку; пропонує ширший вибір сервісів, ніж у конкурентів;
- Azure від Microsoft займає міцне друге місце та є популярним серед розробників C-подібних мов через довготривалі відносини із постачальником послуг;
- Google Cloud від однойменного провайдера посідає третю сходинку та виокремлюється експертизою в технологіях у відкритому доступі, особливо контейнерах завдяки центральній ролі у розробці Kubernetes; інженери компанії тісно взаємодіють із клієнтами.

Усі три постачальники пропонують дуже схожі сервіси, які включають сховища даних, хмарні обчислення, мережеву взаємодію, можливості масштабування, розгортання та управління ресурсами, рішення із захисту інформації тощо [4]. Всі їх сервіси умовно поділяють за ступенем контролю інфраструктури клієнтом: IaaS (інфраструктура як сервіс – найбільший контроль, прикладом є віртуальна машина), PaaS (платформа як сервіс – користувачу надається свобода побудови програмної логіки, але контроль над

обчислювальною машиною майже повністю відсутній; прикладом є лямбда-функції AWS) та SaaS (програмне забезпечення як сервіс – готовий програмний продукт; прикладом є Gmail). Останній підтип не вимагає значних зусиль з боку розробника та дозволяє швидко та ефективно побудувати рішення з наявних інструментів. Застосунок такого типу було розроблено для створення функціоналу оповіщень пацієнтів, а саме з типом Logic App від Microsoft Azure.

Logic App – це платформа для розробки автоматизованих бізнес-процесів та інтеграції застосунків, даних та систем, серед яких є сервіси Azure (Blob Storage, Service Bus), складові пакету Office (Outlook, Excel, SharePoint), сервери баз даних (SQL, Oracle), системи планування ресурсів (SAP, IBM MQ) тощо [6]. Розроблене рішення працює на основі часового тригеру: кожного дня о дев’ятій годині ранку відправляється SQL-запит до БД на вибірку пацієнтів, яким призначено вправи для виконання у цей день, після чого формується SMS-повідомлення на зазначений ними при реєстрації номер, куди за допомогою підключеного стороннього сервісу Twilio відправляється перелік вправ разом з кількістю повторень для кожної.

РОЗДІЛ 4. ОЦІНКА РЕЗУЛЬТАТІВ ТА РОЗГОРТАННЯ

4.1. Тестування

Невід'ємною складовою життєвого циклу розробки програмного забезпечення є проведення тестування. Фундаментальний процес перевірки наявності функціональних та логічних помилок під час різних фаз побудови системи дозволяє зменшити ризик нераціонального використання ресурсів, підвищити якість створюваного продукту та надійність кінцевого рішення.

Виділяють декілька основних типів тестування програмного забезпечення:

- **Димове тестування** перевіряє базову функціональність системи та дозволяє ідентифікувати очевидні помилки на ранніх етапах розробки за рахунок проведення швидких перевірок коректності роботи значущих компонент; якщо етап не пройдено, подальша валідація не є обґрунтованою;
- **Модульне тестування** передбачає створення автоматизованих тестів на найнижчому ізольованому рівні методів класів та функцій; зазвичай є складовою процесу безперервної інтеграції (англ. Continuous Integration);
- **Тестування кінцевої взаємодії** симулює поведінку користувача в повнофункціональній середі виконання, що може включати як прості (вхід в систему), так і комплексні сценарії (верифікація оплати рахунку);
- **Функціональне** або ж **валідаційне тестування** перевіряє результати роботи програмного засобу на відповідність поставленим до нього бізнес-вимогам; у більшості випадків не враховує проміжні стани;
- **Інтеграційне тестування** робить акцент на взаємодії між компонентами: до прикладу валідуватись може коректність роботи бази даних у зв'язці з сервісом обробки замовлень;
- **Тестування на продуктивність** аналізує поведінку системи під навантаженням; для проведення тестів такого типу використовуються

спеціалізовані програми для одночасного надсилання великої кількості запитів (наприклад JMeter, Fiddler, Artillery);

З огляду на особливості розробленої системи, було проведено ряд тестувань з переліку: для окремих компонент створено модульні UNIT-тести, комплексно перевірено взаємодію інструменту діагностики та веб-порталу, протестована робота з базою даних та інтеграція зі сторонніми сервісами; обмеженій кількості осіб було надано доступ до веб-додатку. Усі перераховані міри посприяли покращенню досвіду користування (User Experience), дозволили виправити наявні неочевидні помилки та в цілому підвищили якість продукту. Останнім етапом було проведено валідаційне тестування за участі кандидата біологічних наук Горбунова Олега Андрійовича. Отримані результати засвідчили відповідність системи поставленим до неї вимогам.

4.2. Публікація

В загальному розумінні програмні рішення можуть бути поставлені одним із двох наступних способів: з використанням хмарних сервісів (англ. Cloud) та без (англ. On-Premise). У першому випадку укладається договір з постачальником хмарних послуг, після чого можливим стає оренда інфраструктури, застосування готових інструментів та попередньо встановлених програм: фізичне розгортання на стороні замовника не відбувається. У протилежному випадку усі ресурси та необхідне для них програмне забезпечення закупається та встановлюється безпосередньо в організації клієнта. Обидва підходи мають свої переваги та недоліки. Аргументами на користь підходу On-Premise є захищений доступ до даних та повний контроль над системою, в той же час як хмарні сервіси надають можливість швидкого масштабування, гнучкості та ефективного використання ресурсів, видалення їх у разі необхідності.

Побудована система відноситься до кафедральних розробок Київського національного університету імені Тараса Шевченка, в наявності у якого є низка фізичних серверів. Після консультації з технічним спеціалістом проекту кандидатом фізико-математичних наук Дерев’янченко Олександром Валерійовичем, було прийнято рішення відносно того, що за вимогою уповноважених осіб публікація буде здійснена саме на один із таких серверів. З огляду на це, створено відповідний інструментарій, а саме образ Docker-контейнера, в якому інкапсульовано компоненти системи разом із усіма залежними пакетами.

ВИСНОВКИ

Розглянуто існуючі на ринку рішення з предметної області, де не знайдено аналогів, чим підтверджено актуальність розробки. Проаналізовано актуальні підходи у реалізації програмних засобів, побудові архітектурного дизайну та проведенні тестування технічних продуктів. Наведено огляд сучасних спеціалізованих технологій програмування, визначено їх основні сфери застосування, використано останні практики.

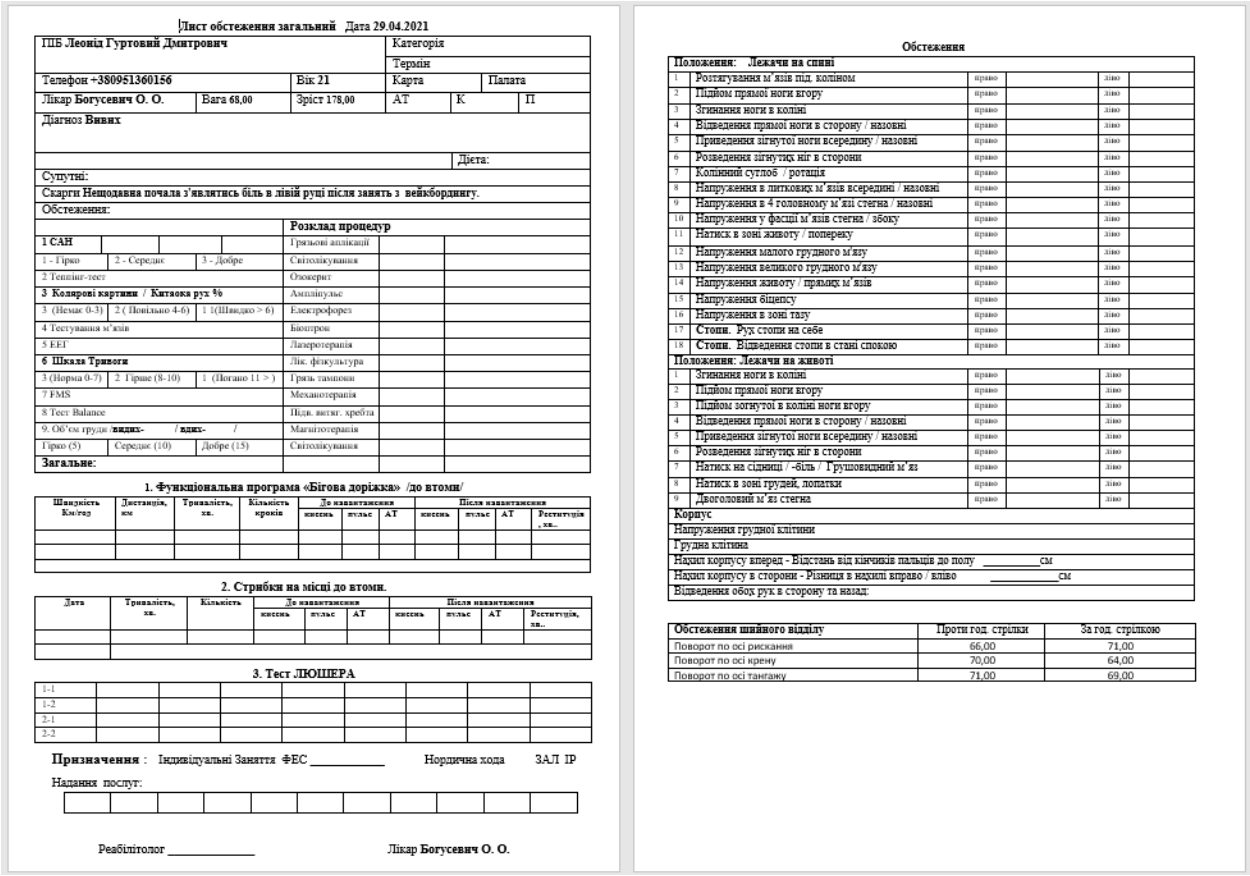
Спроектовано базу даних для збереження інформації про користувачів, пацієнтів, лікарів, дослідження, відділення та допоміжних компонентів, розроблено універсальну, легко розширювальну систему для управління пацієнтами, лікарями, проведення досліджень, генерації звітів та складання інтерактивних планів лікування, забезпечено інструментарій для швидкого розгортання системи, проведено модульне, інтеграційне та валідаційне тестування.

Перспективами розвитку програмного рішення є розширення інструменту діагностики для забезпечення можливості проведення обстежень усіх відділів опорно-рухової системи, створення мобільного додатку для покращення досвіду користування з боку пацієнтів та розробка алгоритмів прогнозування розвитку хвороби на основі отриманої під час досліджень медичної статистики.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Терещенко В.М. Застосування систем візуального моделювання у медицині. // Математичні машини і системи, – 2013, – № 2, – с. 188-194.
2. Tereshchenko, V., Budjak, I., & Fisunenko, A. (2013). The Unified Algorithmic Platform for Solving Complex Problems of Computational Geometry. In Lecture Notes in Computer Science (pp. 424–428). Springer Berlin Heidelberg. doi:10.1007/978-3-642-39958-9_39
3. Popular Database Management Systems | Overview [Електронний ресурс] – Режим доступу до ресурсу: https://support.dbconvert.com/hc/en-us/articles/203189021-Popular-Database-Management-Systems-Overview.
4. AWS vs Azure vs Google Cloud: What's the best cloud platform for enterprise? [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.computerworld.com/article/3429365/aws-vs-azure-vs-google-whats-the-best-cloud-platform-for-enterprise.html.
5. Martin R. C. Screaming Architecture [Електронний ресурс] / Robert C. Martin. – 2011. – Режим доступу до ресурсу: https://blog.cleancoder.com/uncle-bob/2011/09/30/Screaming-Architecture.html.
6. 3DDFA [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/cleardusk/3DDFA.
7. Quintana, M., Karaoglu, S., Alvarez, F., Menendez, J. M., & Gevers, T. (2019). Three-D Wide Faces (3DWF): Facial Landmark Detection and 3D Reconstruction over a New RGB⁻D Multi-Camera Dataset. Sensors (Basel, Switzerland), 19(5), 1103. doi:10.3390/s19051103
8. Schneider, D.C., & Eisert, P. (2009). Fitting a Morphable Model to Pose and Shape of a Point Cloud. VMV.
9. What is Azure Logic Apps [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/azure/logic-apps/logic-apps-overview.
10. Ноубл, Дж., Андерсон, Т., Брэйтуэйт, Г., Казарио, М., Третола, Р. Flex 4. Рецепты программирования. — БХВ-Петербург, 2011. — С. 548. — 720 с.

ДОДАТОК А



Лист обстеження загальний Дата 29.04.2021

ПІБ Леонід Гуртовий Дмитрович				Категорія		
				Термін		
Телефон +380951360156		Вік 21		Карта	Палата	
Лікар **Богусевич О. О.**	Вага 68,00	Зріст 178,00		АТ	К	П
Діагноз **Вивих**						
				Дієта:		
Супутні:						
Скарги Нещодавна почала з'являтись біль в лівій руці після занять з вейкбордингу.						
Обстеження:						

1 САН				Грязьові аплікації	
1 - Гірко	2 - Середнє	3 - Добре		Світолікування	
2 Теппінг-тест				Озокерит	
3 Колярові картини / Китаока рух %				Амплітуль	
3 (Немає 0-3)	2 (Повільно 4-6)	1 1(Швидко > 6)		Електрофорез	
4 Тестування м'язів				Біопатрон	
5 ЕЕГ				Лазеротерапія	
6 Шкала Тривоги				Лік. фізкультура	
3 (Норма 0-7)	2 Гірше (8-10)	1 (Погано 11 >)		Грязь тампони	
7 FMS				Механотерапія	
8 Тест Balance				Підв. витяг. хребта	
9. Об'єм груди /**видих-**		**/ вдих-**	/	Магнітотерапія	
Гірко (5)	Середнє (10)	Добре (15)		Світолікування	
Загальне:					

Розклад процедур

1. Функціональна програма «Бігова доріжка» /до втоми/

Швидкість Км/год	Дистанція, км	Тривалість, хв.	Кількість кроків	До навантаження			Після навантаження			
				кисень	пульс	АТ	кисень	пульс	АТ	Реститутція , хв..

2. Стрибки на місці до втоми.

Дата	Тривалість, хв.	Кількість	До навантаження			Після навантаження			
			кисень	пульс	АТ	кисень	пульс	АТ	Реститутція, хв..

3. Тест ЛЮШЕРА

1-1								
1-2								
2-1								
2-2								

Призначення : Індивідуальні Заняття ФЕС _____ Нордична хода ЗАЛ ІР

Надання послуг:

Реабілітолог _____ Лікар **Богусевич О. О.**

Обстеження

Положення: Лежачи на спині					
1	Розтягування м'язів під. коліном	право		ліво	
2	Підйом прямої ноги вгору	право		ліво	
3	Згинання ноги в коліні	право		ліво	
4	Відведення прямої ноги в сторону / назовні	право		ліво	
5	Приведення зігнутої ноги всередину / назовні	право		ліво	
6	Розведення зігнутих ніг в сторони	право		ліво	
7	Колінний сутлоб / ротація	право		ліво	
8	Напруження в литкових м'язів всередині / назовні	право		ліво	
9	Напруження в 4 головному м'язі стегна / назовні	право		ліво	
10	Напруження у фасції м'язів стегна / збоку	право		ліво	
11	Натиск в зоні животу / попереку	право		ліво	
12	Напруження малого грудного м'язу	право		ліво	
13	Напруження великого грудного м'язу	право		ліво	
14	Напруження животу / прямих м'язів	право		ліво	
15	Напруження біцепсу	право		ліво	
16	Напруження в зоні тазу	право		ліво	
17	**Стопи.** Рух стопи на себе	право		ліво	
18	**Стопи.** Відведення стопи в стані спокою	право		ліво	
Положення: Лежачи на животі					
1	Згинання ноги в коліні	право		ліво	
2	Підйом прямої ноги вгору	право		ліво	
3	Підйом зогнутої в коліні ноги вгору	право		ліво	
4	Відведення прямої ноги в сторону / назовні	право		ліво	
5	Приведення зігнутої ноги всередину / назовні	право		ліво	
6	Розведення зігнутих ніг в сторони	право		ліво	
7	Натиск на сідниці / -біль / Грушовидний м'яз	право		ліво	
8	Натиск в зоні грудей, лопатки	право		ліво	
9	Двоголовий м'яз стегна	право		ліво	
Корпус					
Напруження грудної клітини					
Грудна клітина					
Нахил корпусу вперед - Відстань від кінчиків пальців до полу _____см					
Нахил корпусу в сторони - Різниця в нахилі вправо / кліво _____см					
Відведення обох рук в сторону та назад:					

Обстеження шийного відділу	Проти год. стрілки	За год. стрілкою
Поворот по осі рискання	66,00	71,00
Поворот по осі крену	70,00	64,00
Поворот по осі тангажу	71,00	69,00

Додаток А. Лист обстеження пацієнта