Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему

«Система автоматизації процесів провайдера безперервного професійного розвитку медичних працівників»

Виконав студент 2-го курсу гр. АКІТ-22мз
Спеціальності 151 – Автоматизація та комп'ютерно-інтегровані технології
Сергій ЧОРНОКНИЖНИЙ

Керівник: к.т.н., проф. каф. АІІТ
Євген ПАЛАМАРЧУК
«17» червня 2024 р.

Опонент: к.т.н., доц. каф. КН
Ігор АРСЕНЮК
«18» червня 2024 р.

Допущено до захисту
Завідувач кафедри АІІТ
д.т.н., проф. Олег БІСІКАЛО
«19» 06 2024 року

Вінниця ВНТУ – 2024 рік

Вінницький національний технічний університет

Факультет інтелектуальних інформаційних технологій та автоматизації

Кафедра автоматизації та інтелектуальних інформаційних технологій

Рівень вищої освіти II-ий (магістерський)

Галузь знань 15 – Автоматизація та приладобудування

Спеціальність 151 – Автоматизація та комп'ютерно-інтегровані технології

Освітньо-професійна програма Інтелектуальні комп'ютерні системи

ЗАТВЕРДЖУЮ
Завідувач кафедри АІІТ
д.т.н., проф. Бісікало О. В.
«12» 03 2024 р.

ЗАВДАННЯ

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Чорнокнижному Сергію Іллічу
(ПІБ автора повністю)

1. Тема роботи: Система автоматизації процесів провайдера безперервного професійного розвитку медичних працівників.

Керівник роботи: к.т.н., проф. каф. АІІТ Євген ПАЛАМАРЧУК

Затверджені наказом ВНТУ від «11» Березня 2024 року № 81

2. Строк подання роботи студентом: до «13» червня 2024 року.

3. Вихідні дані до роботи: Монолітна архітектура, безперервний професійний розвиток, БПР, провайдер безперервного професійного розвитку, автоматизація.

4. Зміст текстової частини: Вступ, Загальні відомості про діяльність провайдера БПР, Вибір технології розробки системи автоматизації процесів провайдера БПР, Програмна реалізація системи автоматизації процесів провайдера БПР, Економічний розділ, Висновки, Список використаних джерел.

5. Перелік ілюстративного (або графічного) матеріалу: Взаємозв'язок процесів, які виконує провайдер під час організації та проведення заходу БПР, Діаграма випадків використання, Схема структури бази даних, Структура каталогів та файлів системи.

6. Консультанти розділів магістерської кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийня
1-3	Євген ПАЛАМАРЧУК, к.т.н., проф. каф. АІІТ	12.03.24	17.05.24
4	Володимир КОЗЛОВСЬКИЙ, к.е.н., проф. каф. ЕПВМ	1.05.24	17.05.24

7. Дата видачі завдання: «12» березня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Приміт
1	Аналіз діяльності та процесів провайдерів БПР	12.03 - 31.03.24	вик
2	Аналіз технологій для розробки системи автоматизації процесів провайдера БПР	01.04 - 14.04.24	вик.
3	Програмна реалізація системи автоматизації процесів провайдера БПР	15.04 - 30.04.24	вик.
4	Підготовка економічного розділу	01.05 - 14.05.24	вик.
5	Оформлення пояснювальної записки і графічного матеріалу	15.05 - 29.05.24	вик.
6	Попередній захист роботи	30.05.24	вик.
7	Остаточний захист роботи	20.06.24	вик.

Студент (підпис) Сергій ЧОРНОКНИЖНИЙ (Ім'я та ПРІЗВИЩЕ)

Керівник роботи (підпис) Євген ПАЛАМАРЧУК (Ім'я та ПРІЗВИЩЕ)

АНОТАЦІЯ

УДК 004.4

Чорнокнижний С. І. Система автоматизації процесів провайдера безперервного професійного розвитку медичних працівників. Магістерська кваліфікаційна робота зі спеціальності 151 - Автоматизація та комп'ютерно-інтегровані технології, освітня програма - Інтелектуальні комп`ютерні системи управління. Вінниця: ВНТУ, 2024. 117 с.

На укр. мові Бібліогр.: 54 назв; рис.: 26; табл.: 6.

Метою роботи є розробка системи автоматизації процесів провайдера безперервного професійного розвитку медичних працівників.

Проаналізовано існуючі методи, підходи, а також інструменти для організації автоматизації процесів провайдера безперервного професійного розвитку медичних працівників.

У даній магістерській кваліфікаційній роботі розглянуто основні аспекти розроблення системи автоматизації процесів провайдера безперервного професійного розвитку медичних працівників. Проаналізовано архітектуру системи, розглянуто вимоги до інтерфейсу користувача. В результаті роботи було представлено систему автоматизації процесів провайдера безперервного професійного розвитку медичних працівників.

Ключові слова: монолітна архітектура, безперервний професійний розвиток, БПР, провайдер безперервного професійного розвитку, автоматизація.

ANNOTATION

УДК 004.4

Chornoknyzhnyi S. I. The process automation system of the provider of continuous professional development of medical workers. Master's qualification thesis on specialty 151 - Automation and computer-integrated technologies, educational program - Intelligent computer control systems. Vinnytsia: VNTU, 2024. 117 p.

In Ukrainian language Bibliography: 54 titles; Fig.: 26; tab.: 6.

The purpose of the work is to develop a process automation system for a provider of continuous professional development of medical workers.

The existing methods, approaches, as well as tools for organizing the automation of the processes of the provider of continuous professional development of medical workers have been analyzed.

In this master's qualification work, the main aspects of the development of the process automation system of the provider of continuous professional development of medical workers are considered. The system architecture was analyzed, and the requirements for the user interface were considered. As a result of the work, the process automation system of the provider of continuous professional development of medical workers was presented.

Keywords: monolithic architecture, continuous professional development, BPR, provider of continuous professional development, automation.

ЗМІСТ

ЗМІСТ 6
ВСТУП 8
1. ЗАГАЛЬНІ ВІДОМОСТІ ПРО ДІЯЛЬНІСТЬ ПРОВАЙДЕРА БЕЗПЕРЕРВНОГО ПРОФЕСІЙНОГО РОЗВИТКУ МЕДИЧНИХ ПРАЦІВНИКІВ 11
1.1 Загальна характеристика системи безперервного професійного розвитку медичних працівників 11
1.2 Характеристика основних процесів провайдера безперервного професійного розвитку медичних працівників 14
1.3 Огляд існуючих рішень для автоматизації процесів провайдера безперервного професійного розвитку медичних працівників 23
1.4 Обґрунтування необхідності розробки системи автоматизації процесів провайдера безперервного професійного розвитку медичних працівників 24
1.5 Висновки до розділу 1 26
2. ВИБІР ТЕХНОЛОГІЇ РОЗРОБКИ СИСТЕМИ АВТОМАТИЗАЦІЇ ПРОЦЕСІВ ПРОВАЙДЕРА БЕЗПЕРЕРВНОГО ПРОФЕСІЙНОГО РОЗВИТКУ МЕДИЧНИХ ПРАЦІВНИКІВ 28
2.1 Вибір архітектури системи 28
2.2 Вибір мови програмування 34
2.3 Вибір системи управління базами даних 42
2.4 Вибір технологій для побудови інтерфейсу користувача 48
2.5 Висновки до розділу 2 54
3. ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ АВТОМАТИЗАЦІЇ ПРОЦЕСІВ ПРОВАЙДЕРА БЕЗПЕРЕРВНОГО ПРОФЕСІЙНОГО РОЗВИТКУ МЕДИЧНИХ ПРАЦІВНИКІВ 56
3.1 Опис процесів в системі 56
3.2 Розробка структури бази даних 69

3.3 Розробка архітектури системи 79
3.3 Розробка інтерфейсу користувача 85
3.5 Висновки до розділу 3 91
4. ЕКОНОМІЧНИЙ РОЗДІЛ 94
4.1 Технологічний аудит розробленої системи автоматизації процесів, що їх виконує провайдер під час організації та проведення заходів безперервного професійного розвитку медичних працівників .. 94
4.2 Розрахунок витрат на розроблення системи автоматизації процесів провайдера безперервного професійного розвитку медичних працівників 98
4.3 Розрахунок економічного ефекту від можливої комерціалізації нашої розробки 102
4.4 Висновки до розділу 4 108
ВИСНОВКИ 109
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 112
ДОДАТКИ 118
Додаток А (обов'язковий) 119
Додаток Б (обов'язковий) 122
Додаток В (обов'язковий) 136
Додаток Г (довідниковий) 137

ВСТУП

Актуальність. В Україні з 1 січня 2022 року набрала чинності Постанова КМУ від 14 липня 2021 року № 725 «Про затвердження Положення про систему безперервного професійного розвитку медичних та фармацевтичних працівників» [1]. Відповідно до вимог вказаної постанови, МОЗ України запровадило нову систему безперервного професійного розвитку для лікарів (БПР), згідно якої працівники сфери охорони здоров'я після здобуття вищої освіти та отримання сертифіката лікаря-спеціаліста, провізора-спеціаліста або диплома про закінчення закладу фахової передвищої освіти зобов'язані здійснювати безперервний професійний розвиток [2].

На сучасному світовому ринку не представлено рішень для автоматизації процесів провайдера безперервного професійного розвитку медичних працівників, оскільки система БПР є специфічною для України.

На українському ринку також не представлено opensource рішень для автоматизації процесів провайдера безперервного професійного розвитку медичних працівників.

Деякі варіанти рішень були розроблені під замовлення великих провайдерів для застосування у власній мережі.

Для малих або середніх провайдерів подібних аналогів на українському ринку немає, а дозволити собі розробку або покупку рішень, які розраховані на великих провайдерів вони не можуть, оскільки це досить дорого і використовувати такі рішення невигідно з боку бюджету таких провайдерів.

Таким чином, буде актуальним та доцільним розробка бюджетного цілісного рішення, яке задовольнить потреби в автоматизації процесів провайдера безперервного професійного розвитку медичних працівників.

Метою даної роботи є розробка системи автоматизації процесів провайдера безперервного професійного розвитку медичних працівників.

Для того, щоб досягнути вищевказаної мети потрібно розв’язати низку таких **задач**:

- розглянути основні процеси в діяльності провайдера безперервного професійного розвитку медичних працівників;

- проаналізувати технології та підходи для програмної реалізації системи автоматизації процесів провайдера безперервного професійного розвитку медичних працівників;

- на основі розглянутих підходів та технологій програмно реалізувати систему автоматизації процесів провайдера безперервного професійного розвитку медичних працівників.

Об’єктом дослідження є процеси в діяльності провайдера безперервного професійного розвитку медичних працівників.

Предметом дослідження є діяльність провайдера безперервного професійного розвитку медичних працівників.

Методи дослідження. Дослідження наукової літератури та існуючих систем для розуміння їхніх переваг і недоліків. Порівняння функціоналу, архітектури та інших аспектів існуючих систем. Контрольоване спостереження та тестування для вивчення причинно-наслідкових зв’язків.

Основні науково-технічні результати полягають у розробці та впровадженні нової системи автоматизації процесів провайдера безперервного професійного розвитку медичних працівників.

Практична цінність. Проведені дослідження надають змогу впроваджувати систему автоматизації процесів провайдера безперервного професійного розвитку медичних працівників. Це сприяє оптимізації діяльності провайдера безперервного професійного розвитку медичних працівників.

Апробація. Результати теоретичних і експериментальних досліджень викладені у доповіді на науково-технічній конференції факультету інтелектуальних інформаційних технологій та автоматизації - LIII

Всеукраїнська науково-технічна конференція факультету інтелектуальних інформаційних технологій та автоматизації у 2024 році [55].

Результати роботи були впроваджені в роботу діючого провайдера безперервного професійного розвитку медичних працівників Товариство з обмеженою відповідальністю «Агенція безперервного професійного розвитку». Акт впровадження № 1 від 04.06.2024, продемонстровано в додатку Г.

1. ЗАГАЛЬНІ ВІДОМОСТІ ПРО ДІЯЛЬНІСТЬ ПРОВАЙДЕРА БЕЗПЕРЕРВНОГО ПРОФЕСІЙНОГО РОЗВИТКУ МЕДИЧНИХ ПРАЦІВНИКІВ

1.1 Загальна характеристика системи безперервного професійного розвитку медичних працівників

В Україні з 1 січня 2022 року набрала чинності Постанова КМУ від 14 липня 2021 року № 725 «Про затвердження Положення про систему безперервного професійного розвитку медичних та фармацевтичних працівників» [1]. Відповідно до вимог вказаної постанови, МОЗ України запровадило нову систему безперервного професійного розвитку для лікарів (БПР), згідно якої працівники сфери охорони здоров'я після здобуття вищої освіти та отримання сертифіката лікаря-спеціаліста, провізора-спеціаліста або диплома про закінчення закладу фахової передвищої освіти зобов'язані здійснювати безперервний професійний розвиток [2].

Стара система підвищення кваліфікації передбачала, що лікар за 1 рік до проходження атестації, яка мала проходити мінімум один раз на 5 років, повинен обов'язково пройти 4-х тижневі курси, які називалися «передатестаційний цикл» (ПАЦ), а в період між проходженням ПАЦ міг фактично не брати жодної участь в будь-яких освітніх заходах. Дане положення речей призводило до того, що була відсутня збалансованість освітніх навантажень і три роки після атестації по суті були часом «відпочинку» від професійного вдосконалення [3].

Нова ж система БПР для медичних працівників передбачає сучасний підхід до професійного вдосконалення, який дозволяє медичному працівнику вільно обирати цікаві для себе освітні заходи, зручний формат та час для навчання, а також виконавців освітніх послуг. Замість проходження ПАЦ, які по суті були формальним навчанням, медичні працівників тепер зможуть постійно розвивати свої професійні компетенції. Також за проходження

навчання передбачено нарахування балів, які потрібно використовувати для атестації. На практиці БПР в Україні впроваджується з квітня 2019 року. З 2020 року БПР став обов'язковим для лікарів, а з 2024 року – для фармацевтичних працівників та молодшого медичного персоналу [4].

Згідно Наказу МОЗ від 22 лютого 2019 року № 446 «Деякі питання безперервного професійного розвитку лікарів» [5], кожний працівник сфери охорони здоров'я зобов'язаний набирати за календарний рік щонайменше 50 балів БПР. Максимальна кількість балів не обмежена, але попередньо враховані бали БПР на майбутні періоди не переносяться.

Бали БПР нараховуються за проходження різних видів навчання, які медичний працівник може вільно обирати на основі запропонованого МОЗ переліку, яких передбачено три види:

1. Формальна освіта.
   - Присвоєння кваліфікації «лікар-спеціаліст» відповідної лікарської спеціальності.
   - Здобуття освітньо-наукового та наукового рівнів вищої освіти галузі знань «Охорона здоров'я» (доктор філософії, доктор наук).
2. Неформальна освіта.
   - Підвищення кваліфікації на циклах тематичного удосконалення в закладах (на факультетах) післядипломної освіти.
   - Навчання або медичне стажування в закладі вищої освіти/закладі охорони здоров'я за межами закладу, де працює фахівець.
3. Інформальна освіта.
   - Науково-практичні конференції, конгреси, симпозіуми (рекламні доповіді не враховуються).
   - Професійний розвиток за дистанційною формою навчання з використанням електронних навчальних ресурсів.
   - Навчання на тренінгах з оволодіння практичними навичками або симуляційних тренінгах.

- Тематичне навчання (семінари, майстер-класи, фахові школи тощо)
- Публікація огляду або статті в журналі з імпакт-фактором [6].

Сертифікати про нарахування балів БПР видаються провайдерами заходів БПР - юридичними особами, які провадять діяльність з організації та проведення заходів БПР.

З метою забезпечення здійснення БПР діє спеціально розроблена електронна система, яка призначена для збереження інформації про провайдерів, заходи БПР та обліку балів БПР.

Власником Системи БПР, у т. ч. програмного забезпечення, є держава в особі МОЗ України.

Адміністратором Системи БПР призначено Державне некомерційне підприємство «Центр тестування професійної компетентності фахівців з вищою освітою напрямів підготовки “Медицина” і “Фармація” при МОЗ України» [7].

Введення спеціалізованої електронної системи забезпечення безперервного професійного розвитку працівників сфери охорони здоров’я планується з 1.04.2024 р. [8]

На даний момент збереження інформації про провайдерів, заходи БПР та обліку балів БПР тимчасово реалізовано з використанням Google Таблиць, в яких створено наступні загально доступні документи:

1. Провайдери заходів БПР [9].
2. Перелік заходів БПР 2024 [10]. Також раніше були створені документи «Перелік заходів БПР 2022» та «Перелік заходів БПР 2023».
3. Бали БПР 2024 [11]. Також раніше були створені документи «Бали БПР 2022» та «Бали БПР 2023».

Перехідний період з Google Таблиць до спеціалізованої електронної системи заплановано у проміжку з 1.04 до 1.07.2024 року, коли будуть застосовуватись обидві системи паралельно.

З 1.07.2024 року планується використання виключно спеціалізованої електронної системи [8].

1.2 Характеристика основних процесів провайдера безперервного професійного розвитку медичних працівників

Відповідно до вимог постанови № 725 [1] юридична особа, яка має намір бути Провайдером заходів БПР за які нараховуються бали, надає Адміністратору наступний пакет документів:

1. Заяву в електронній формі.

2. Витяг із Єдиного державного реєстру юридичних осіб, фізичних осіб – підприємців та громадських формувань із переліком КВЕД.

3. Положення про оцінку заходів безперервного професійного розвитку на ознаки академічної доброчесності та дотримання принципів практичної медичної/фармацевтичної/реабілітаційної діяльності, заснованої на доказах, затверджене Провайдером.

4. Методологію оцінювання набутих знань, компетентностей та практичних навичок працівників сфери охорони здоров’я, що затверджується Провайдером.

5. Положення про запобігання конфлікту інтересів під час проведення заходів безперервного професійного розвитку та недопущення залучення і використання коштів фізичних (юридичних) осіб для реклами лікарських засобів, медичних виробів, допоміжних засобів реабілітації або медичних послуг, затверджене Провайдером.

6. Заяву засновників (учасників, власників), посадових осіб про відсутність конфлікту інтересів [12].

У разі подання повного та коректно оформленого пакету документів протягом 20 робочих днів з дня його подання, Адміністратор присвоює провайдеру реєстраційний номер та вносить його дані до реєстру Провайдерів Системи БПР [9].

Провайдер під час організації та проведення заходу БПР має виконувати наступні операції:

1. Сформувати та викласти на власному сайті програму, навчальну програму та картку заходу
2. Виконати реєстрацію заходу в Центрі тестування МОЗ
3. Провести реєстрацію учасників заходу
4. Забезпечити проведення тестування в електронному вигляді
5. Згенерувати сертифікати за встановленим МОЗ зразком
6. Прозвітувати про проведення заходу перед Центром тестування МОЗ.

Інформація про кожен освітній захід, за проходження якого медичним працівникам мають бути нараховані бали БПР, обов'язково має бути окремо розміщена на офіційному сайті Провайдера.

Картка заходу має містити наступні пункти:

1. Назва заходу БПР
2. Назва Провайдера
3. Співорганізатори заходу
4. Цільова аудиторія
5. Вид заходу БПР
6. Формат заходу БПР
7. Запланована кількість учасників
8. Мета навчання
9. Метод навчання
10. Кількість балів БПР
11. Дата заходу БПР
12. Місце проведення заходу БПР
13. Прізвище, ім'я та по батькові лекторів/тренерів
14. Резюме лекторів/тренерів
15. Програма заходу БПР

16. Опис вимог рівня знань, володіння темою, навичок, досвіду учасників до моменту реєстрації на даний захід

17. Методи оцінювання набутих знань

18. Номер заходу в системі БПР

Законодавством передбачено наступні види заходів БПР:

1. Електронний навчальний курс – підготовлений комплекс навчально-методичних матеріалів та освітніх послуг для проведення індивідуального та групового навчання із застосуванням електронних технологій.

2. Майстер-клас - представлення та демонстрація окремих методик, технологій діагностики, профілактики, лікування та реабілітації для підвищення професійного рівня та/або обміну передовим досвідом серед учасників заходу та залучення їх до новітніх галузей знань.

3. Тренінг з оволодіння практичними навичками або Симуляційний тренінг - набуття кожним з учасників заходу специфічної клінічної та/або практичної навички та/або компетенції в умовах штучного, наближеного до реальності середовища для забезпечення.

4. Тренінг - опанування учасниками заходу певних професійних знань та навичок з окремих розділів спеціальності, і з актуальних питань організації медичної, фармацевтичної, реабілітаційної допомоги та медичного обслуговування за відповідними напрямами.

5. Семінар - набуття нових актуальних знань з організації медичної, фармацевтичної, реабілітаційної допомоги та медичного обслуговування за відповідними напрямами. Також передбачена можливість обговорення отриманої інформації під час проходження навчання у невеликих групах.

6. Фахова або тематична школа - навчання направлене на вивчення актуальних питань відповідної спеціальності, що об’єднує заняття великих груп для засвоєння теоретичної частини під час лекцій та заняття малих груп для проведення семінарів або практичних занять.

7. Наукова та/або науково-практична конференція (у т. ч. конгрес, з’їзд, симпозіум) – особлива форма проведення наукової діяльності у вигляді

зборів або наради фахівців та/або наукових працівників з метою демонстрації результатів своєї дослідницької роботи або результатів аналізу існуючих досягнень медицини та фармації, а також узагальнення та поширення кращого досвіду, створення теоретичних та методичних умов для його впровадження [1].

Заходи можуть бути одноденними та багатоденними (2 та більше днів). Від тривалості заходу та його виду залежить кількість балів БПР, що будуть нараховані за даний курс:

1. Одноденний захід:
   - Семінар, Фахова школа, Майстер-клас, Тренінг - 10 балів
   - Симуляційний тренінг, Тренінг з оволодіння практичними навичками - 15 балів
   - Наукова конференція, Науково-практична конференція, Конгрес, З'їзд, Симпозіум – 5 балів
2. Багатоденний захід:
   - Семінар, Фахова школа, Майстер-клас, Тренінг - 20 балів
   - Симуляційний тренінг, Тренінг з оволодіння практичними навичками - 25 балів
   - Наукова конференція, Науково-практична конференція, Конгрес, З'їзд, Симпозіум – 10 балів [5]

Під цільовою аудиторією розуміється перелік лікарських та фармацевтичних спеціальностей для яких передбачений даний захід. Із повним переліком можна ознайомитися на сайтах деяких провайдерів БПР [13].

Відносно кількості інструкторів та учасників також є певні законодавчі обмеження, які необхідно врахувати:

1. Семінар, Тренінг – не більше 20 учасників на 1 інструктора.
2. Фахова школа – не більше 12 учасників на 1 інструктора.
3. Симуляційний тренінг, Тренінг з оволодіння практичними навичками – не більше 6 учасників на 1 інструктора.

4. Майстер-клас, Наукова конференція, Науково-практична конференція, Конгрес, З’їзд, Симпозіум – без обмежень [1].

За 20 робочих днів до початку заходу Провайдер подає Адміністратору інформацію про заходи за посиланням на Google Форму [14].

У вказаній Google Формі мають бути заповнені наступні дані:

1. Реєстраційний номер провайдера
2. Тема заходу
3. Вид освітнього заходу
4. Формат заходу
5. Акредитація заходу
6. Кількість балів, що нараховуються відповідно до законодавства
7. Дата початку проведення заходу
8. Дата завершення проведення заходу
9. Цільова аудиторія
10. Мова проведення освітнього заходу
11. Місце проведення
12. Посилання для реєстрації на захід
13. Посилання на веб-сайт провайдера, де розміщена інформація про захід
14. ПІБ контактної особи провайдера, яка відповідальна за організацію та проведення заходу
15. Номер телефону контактної особи провайдера, яка відповідальна за організацію та проведення заходу

Також у вказану Google Форму мають бути завантажені Програма та Навчальна програма заходу у форматі doc або docx.

Програма заходу має містити наступні пункти:

1. Вид заходу
2. Тема заходу
3. Дата
4. Час початку

5. Опис структури заходу

6. Час завершення

7. ПІБ Викладача/Тренера

8. Резюме Викладача/Тренера

9. Кількість інструкторів

10. Кількість учасників

Навчальна програма заходу має містити наступні пункти:

1. Тема заходу

2. Вид заходу

3. Цільова аудиторія

4. Мета заходу

5. Перелік компетентностей, що набуваються або вдосконалюються

6. Опис структури заходу

7. Загальний обсяг навчального навантаження

8. Форми та методи організації та проведення заходу

9. Матеріально-технічне забезпечення заходу

10. Форми підсумкового контролю

Також у Google Форму може бути завантажене підтвердження акредитації навчального заходу за кордоном чи в Україні EACCME/ACCME/RCPSC чи сертифікацію ERC/ILCOR/AHA, що потрібно для нарахування учасникам подвійних балів БПР [12].

Адміністратор перевіряє внесену інформацію про захід БПР, на повноту та відповідність законодавству і підтверджує реєстрацію заходу в Системі БПР або відмовляє у реєстрації заходу. В якості підтвердження реєстрації заходу в Системі БПР, Адміністратор присвоює заходу БПР реєстраційний номер та вносить його до реєстру заходів БПР [10].

Кожен освітній захід, за проходження якого медичним працівникам мають бути нараховані бали БПР, потрібно зареєструвати окремо.

Також слід відмітити, що реєстрація заходів БПР є платною і послуги Центру тестування оплачуються по факту виконання реєстрації, незалежно

від того, був проведений захід чи був відмінений, або дані були подані із помилками.

Центр тестування вимагає, щоб всі учасники після завершення курсу обов'язково пройшли тестування в електронному вигляді.

Сертифікати видаються Провайдером після завершення заходу БПР.

Обов'язковим компонентом сертифікату є номер, який виражається у цифрах у наступному форматі: xxxx-zxxx-zxxxxxx-zxxxxx де,

X – будь-яка цифра

Z – будь-яка цифра, окрім 0 (рис. 1.1).

Рисунок 1.1 – Формат номеру сертифікату

Особливістю даного номеру є його унікальність, тобто ні в якому разі не має бути 2-х сертифікатів з однаковими номерами.

Рік проведення заходу БПР – це 4-х цифровий запис року і є однаковим для всіх Заходів у відповідному році.

Реєстраційний номер Провайдер присвоюється Центром тестування, ніколи не змінюється і є однаковим для всіх Заходів відповідного Провайдера.

Реєстраційний номер заходу також присвоюється Центром тестування ніколи не змінюється і є однаковим для всіх сертифікатів відповідного Заходу.

Таким чином, унікальність номера забезпечується за рахунок Реєстраційного номера учасника, тому обов'язок Провайдера забезпечити коректне присвоєння даних реєстраційних номерів та їх унікальність.

Сертифікат обов'язково має містити наступні відомості:

1. Повну юридичну назву Провайдера (відповідно до Єдиного державного реєстру юридичних осіб, фізичних осіб – підприємців та громадських формувань).
2. Вид заходу БПР.
3. Тему заходу БПР.
4. Прізвище, власне ім'я, по батькові (за наявності) отримувача сертифіката.
5. Спеціальності за якими проводилось навчання (цільова аудиторія)
6. Кількість балів БПР.
7. Дата проведення заходу БПР.
8. Номер сертифікату.
9. Підпис та ПІБ керівника Провайдера, який гарантує проведення оцінювання набутих знань та практичних навичок, а також дотримання всіх вимог законодавства під час проведення заходу.

Для сертифікатів, які видаються учасникам електронного навчального курсу, до вище вказаних відомостей додається також кількість навчальних годин та дата видачі [12].

Кількість балів учаснику заходу БПР розраховується згідно з наказами МОЗ [5].

Після завершення заходу, для проведення обліку балів БПР, Провайдер протягом 20 робочих днів має подати Адміністратору дані про нарахування балів заповнивши відповідну Google Форму [15].

У вказаній Google Формі мають бути заповнені наступні дані:

1. Реєстраційний номер провайдера
2. Номер заходу БПР
3. ПІБ контактної особи
4. № телефону (мобільний) контактної особи

Також у вказану Google Форму мають бути завантажені наступні файли:

1. Програма заходу у форматі doc або docx.

2. Копія одного сертифіката заходу (оформленого повністю) у форматі pdf або зображення.
3. Файл "Шаблон Видані сертифікати 2024" у форматі xls або xlsx, відповідно до шаблону розміщеного на сайті Центру тестування.
4. Файл "Контроль успішності курсантів" у форматі xls або xlsx, відповідно до шаблону розміщеного на сайті Центру тестування.

Файл "Шаблон Видані сертифікати 2024" має містити наступні дані:

1. Реєстраційний номер Провайдера
2. Реєстраційний номер заходу
3. Номер сертифіката
4. Прізвище, власне ім'я, по батькові (за наявності) учасника
5. Бали БПР
6. Дата народження
7. Засоби зв’язку (електронна адреса)
8. Освіта
9. Місце роботи
10. Займана посада
11. Результати оцінювання за проходження заходу БПР учасників заходу, які отримали сертифікати

Файл "Контроль успішності курсантів" має містити наступні дані:

1. Дата та час тестування
2. ПІБ курсанта
3. Результати вихідного контролю успішності

Адміністратор перевіряє внесену інформацію про бали БПР, на повноту та відповідність законодавству і вносить їх до реєстру балів БПР [11].

1.3 Огляд існуючих рішень для автоматизації процесів провайдера безперервного професійного розвитку медичних працівників

На сучасному світовому ринку не представлено рішень для автоматизації процесів провайдера безперервного професійного розвитку медичних працівників, оскільки система БПР є специфічною для України.

На українському ринку також не представлено opensource рішень для автоматизації процесів провайдера безперервного професійного розвитку медичних працівників.

Деякі варіанти рішень були розроблені під замовлення великих провайдерів для застосування у власній мережі. Зокрема, власні системи автоматизації використовуються наступними великими провайдерами:

1. Громадська спілка «Безперервного професійного розвитку стоматологів» - 1224 заходи БПР за 2023 рік [16].
2. Товариство з обмеженою відповідальністю «Медвойс» - 399 заходів БПР за 2023 рік [17].
3. Товариство з обмеженою відповідальністю «Професійна платформа» - 279 заходів БПР за 2023 рік [18].
4. Громадська організація «Медична платформа Прогрес» - 148 заходів БПР за 2023 рік [19].
5. Товариство з обмеженою відповідальністю «Група компаній МедЕксперт» - 143 заходів БПР за 2023 рік [20].
6. Громадська організація «Агенція екстреної медичної допомоги» - 93 заходи БПР за 2023 рік [21].

Для малих або середніх провайдерів подібних аналогів на українському ринку немає, а дозволити собі розробку або покупку рішень, які розраховані на великих провайдерів вони не можуть, оскільки це досить дорого і використовувати такі рішення невигідно з боку бюджету таких провайдерів.

1.4 Обґрунтування необхідності розробки системи автоматизації процесів провайдера безперервного професійного розвитку медичних працівників

Таким чином, буде актуальним та доцільним розробка бюджетного цілісного рішення, яке задовольнить потреби в автоматизації процесів провайдера безперервного професійного розвитку медичних працівників.

Дії провайдера під час організації та проведення заходу БПР проходять через наступні етапи:

1. Формування та розміщення на власному сайті програми, навчальної програми та картки заходу
2. Виконання реєстрації заходу в Центрі тестування МОЗ
3. Проведення реєстрації учасників заходу
4. Забезпечення проведення тестування в електронному вигляді
5. Генерація сертифікатів за встановленим МОЗ зразком
6. Подання звіту про проведення заходу в Центр тестування МОЗ.

Взаємозв’язок даних процесів можна представити у вигляді блок-схеми, наведеної на рис. 1.2.

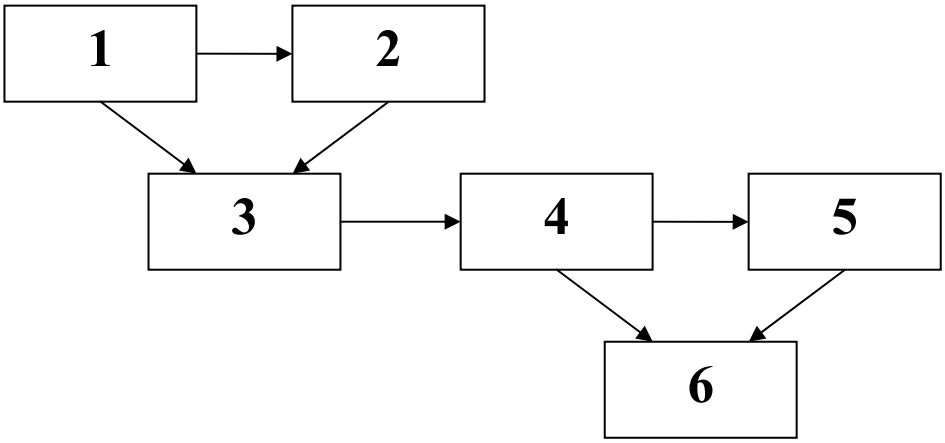


Рисунок 1.2 – Взаємозв’язок процесів, які виконує провайдер під час організації та проведення заходу БПР

Сформулюємо постановку задачі, щодо автоматизації процесів, які виконує провайдер під час організації та проведення заходу БПР на кожному з етапів.

1. Формування та розміщення на власному сайті програми, навчальної програми та картки заходу

- Розробка інтерфейсу, в якому працівники провайдера зможуть вводити в систему всі необхідні дані про захід БПР, згідно вимог постанови №725 та Центру тестування МОЗ.
- Розробка розділу, в якому всім відвідувачам сайту буде відображатися інформація про майбутні заходи БПР – детальний опис, програма та картка заходу.

2. Виконання реєстрації заходу в Центрі тестування МОЗ

- Розробка інтерфейсу, в якому працівники провайдера зможуть на основі даних, введених в систему, сформувати програму, навчальну програму, згідно вимог постанови №725 та Центру тестування МОЗ.

3. Проведення реєстрації учасників заходу

- В розділі, де розміщена інформація про заходи БПР, потрібно передбачити інструменти за допомогою яких відвідувач зможе зареєструватися на захід БПР.
- Передбачити інтерфейс, в якому працівники провайдера зможуть самостійно вводити в систему нових учасників заходу БПР.

4. Забезпечення проведення тестування в електронному вигляді

- Розробка інтерфейсу, в якому працівники провайдера зможуть вводити в систему бази тестів з прикріпленням їх до відповідного заходу БПР.
- Передбачити інструмент за допомогою яких учасник заходу БПР зможе пройти тестування в дні проведення заходу. До початку заходу та після його проведення тестування не має бути доступним.

- Інструмент тестування має забезпечити вибірку необхідної кількості випадкових запитань з бази тестів та розмістити їх у випадковому порядку. Також має бути реалізовано лімітування часу на відповіді та захист від спроб фальсифікації результатів. Також інструмент після тестування має показувати помилкові відповіді для роботи над помилками.

5. Генерація сертифікатів за встановленим МОЗ зразком
   - Розробка інтерфейсу, в якому працівники провайдера зможуть на основі даних учасників заходу БПР, введених в систему, згенерувати сертифікати, згідно вимог постанови №725 та Центру тестування МОЗ.
   - Інструмент має передбачати генерацію унікального номера для кожного сертифікату, згідно вимог постанови №725 та Центру тестування МОЗ.
6. Подання звіту про проведення заходу в Центр тестування МОЗ.
   - Розробка інтерфейсу, в якому працівники провайдера зможуть на основі даних, введених в систему, сформувати файли для подання звіту до Центру тестування МОЗ – Шаблон обліку балів БПР, результати тестування та програму заходу.

1.5 Висновки до розділу 1

На сучасному світовому ринку не представлено рішень для автоматизації процесів провайдера безперервного професійного розвитку медичних працівників, оскільки система БПР є специфічною для України.

На українському ринку також не представлено opensource рішень для автоматизації процесів провайдера безперервного професійного розвитку медичних працівників.

Деякі варіанти рішень були розроблені під замовлення великих провайдерів для застосування у власній мережі.

Для малих або середніх провайдерів подібних аналогів на українському ринку немає, а дозволити собі розробку або покупку рішень, які розраховані на великих провайдерів вони не можуть, оскільки це досить дорого і використовувати такі рішення невигідно з боку бюджету таких провайдерів.

Дії провайдера під час організації та проведення заходу БПР проходять через наступні етапи:

1. Формування та розміщення на власному сайті програми, навчальної програми та картки заходу
2. Виконання реєстрації заходу в Центрі тестування МОЗ
3. Проведення реєстрації учасників заходу
4. Забезпечення проведення тестування в електронному вигляді
5. Генерація сертифікатів за встановленим МОЗ зразком
6. Подання звіту про проведення заходу в Центр тестування МОЗ.

Таким чином, буде актуальним та доцільним розробка бюджетного цілісного рішення, яке задовольнить потреби в автоматизації процесів провайдера безперервного професійного розвитку медичних працівників.

2. ВИБІР ТЕХНОЛОГІЇ РОЗРОБКИ СИСТЕМИ АВТОМАТИЗАЦІЇ ПРОЦЕСІВ ПРОВАЙДЕРА БЕЗПЕРЕРВНОГО ПРОФЕСІЙНОГО РОЗВИТКУ МЕДИЧНИХ ПРАЦІВНИКІВ

2.1 Вибір архітектури системи

Локальні застосунки можуть включати в себе різні сервісні програми, утиліти, графічні та текстові редактори тощо. Розробка розширених інформаційних систем обумовлює те, що вони не можуть складатися з окремих компонентів, які не пов'язані між собою.

Відповідно, сучасні інформаційні системи розробляються з використанням різних розподілених архітектур:

- Монолітна архітектура
- Клієнт-серверна архітектура
- Розподілена архітектура
- Мікросервісна архітектура [22]

Монолітна архітектура - це традиційна модель побудови програмного забезпечення, яка являє собою єдиний модуль, що працює повністю автономно та незалежно від інших модулів (рис. 2.1). Дана архітектура є найбільш простою та поширеною і може бути реалізована на різних мовах програмування, таких як Java, PHP, JavaScript або Python [23].

Монолітна архітектура має певні переваги.

По-перше – це простота розробки: Монолітні програми легше розробляти, тому що немає необхідності керувати безліччю окремих сервісів або модулів.

Друга перевага – це зручність розгортання: Всі компоненти системи упаковані в один пакет, що спрощує процес розгортання.

Третя перевага – це простота масштабування: Монолітну систему можна масштабувати, розмістивши її за балансувальником навантаження та запустивши декілька копій.

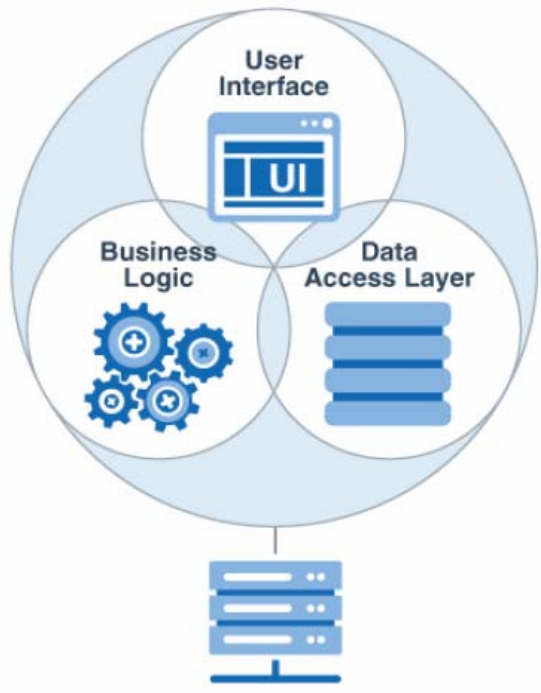


Рисунок 2.1 – Компоненти монолітної архітектури

Також до переваг слід віднести наскрізні тести: У монолітній системі простіше проводити наскрізні тести, оскільки всі її елементи знаходяться в одному пакеті [24].

Разом з тим монолітна архітектура має і певні недоліки.

В першу чергу – це складність підтримки: З часом розвитку моноліт може стати дуже великим та складним, що ускладнює додавання нового функціоналу та пошук помилок.

Разом з простотою масштабування виникають також певні проблеми: Якщо тільки одній частині системи необхідні додаткові ресурси, то масштабувати тільки її, не задіявши інші частини, не можливо.

Також важливим недоліком є відсутність ізоляції: Помилка в одній з частин програми може вивести з ладу всю систему, оскільки немає розділення між компонентами.

Ще до недоліків слід віднести проблеми з оновленням: Оновлення бібліотек або технологій в монолітній системі може бути досить проблематичним, оскільки це необхідно виконувати одночасно для всієї системи [25].

Монолітну архітектуру рекомендується використовувати для невеликих, простих додатків, які не потребують високої масштабованості та гнучкості.

Клієнт-серверна архітектура – це модель побудови інформаційних систем, де певні завдання розподілені між серверами та клієнтами. Клієнти надсилають запити до сервера, який передає необхідні дані (рис. 2.2).

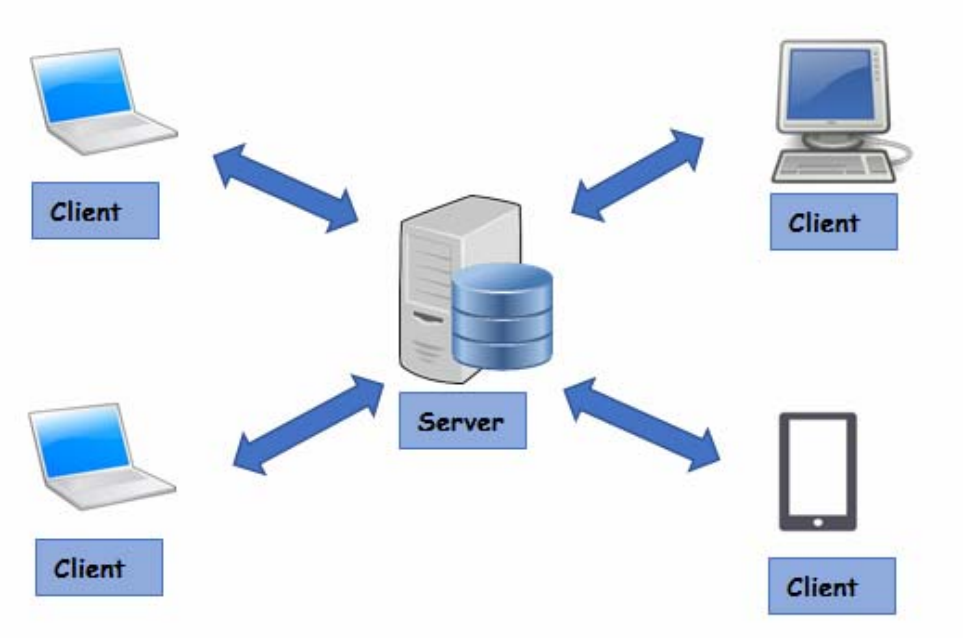


Рисунок 2.2 – Компоненти клієнт-серверної архітектури

Як і будь яка система, клієнт-серверна архітектура має певні переваги.

В першу чергу, це – централізоване зберігання даних: Всі дані централізовано зберігаються на сервері, що полегшує управління та забезпечення безпеки.

Друга перевага - це масштабованість: Сервер обслуговує велику кількість клієнтів, та, за необхідності його можна масштабувати шляхом збільшення апаратних ресурсів.

І нарешті – поділ відповідальності: Логіка програми розподілена між сервером та клієнтом, що дає можливість оптимізувати використання ресурсів [26].

Але у клієнт-серверної архітектури також є і недоліки.

Перше – це залежність від сервера: Робота всієї системи залежить від сервера, і якщо сервер вийде з ладу, то система не буде працювати, навіть при збереженні роботи.

Також важливий недолік – це вразливість до атак: Централізоване зберігання даних робить сервер метою для атак, і зламування сервера може спричинити виток даних.

Ну і нарешті, складність масштабування: Дану систему можна масштабувати, але це вимагатиме суттєвих інвестицій в серверне обладнання.

Дана модель широко використовується в різних сервісах до, і вибір між даною архітектурою та іншими моделями, залежить від конкретних умов та вимог використання [27].

Клієнт-серверна архітектура рекомендована для додатків середніх розмірів з високою гнучкістю та масштабованістю – CMS, електронні магазини, СУБД, соціальні мережі тощо.

Розподілена архітектура - це така модель інформаційної системи, в якій компоненти розміщені на різних серверах і взаємодіють між собою для досягнення спільної мети (рис. 2.3).

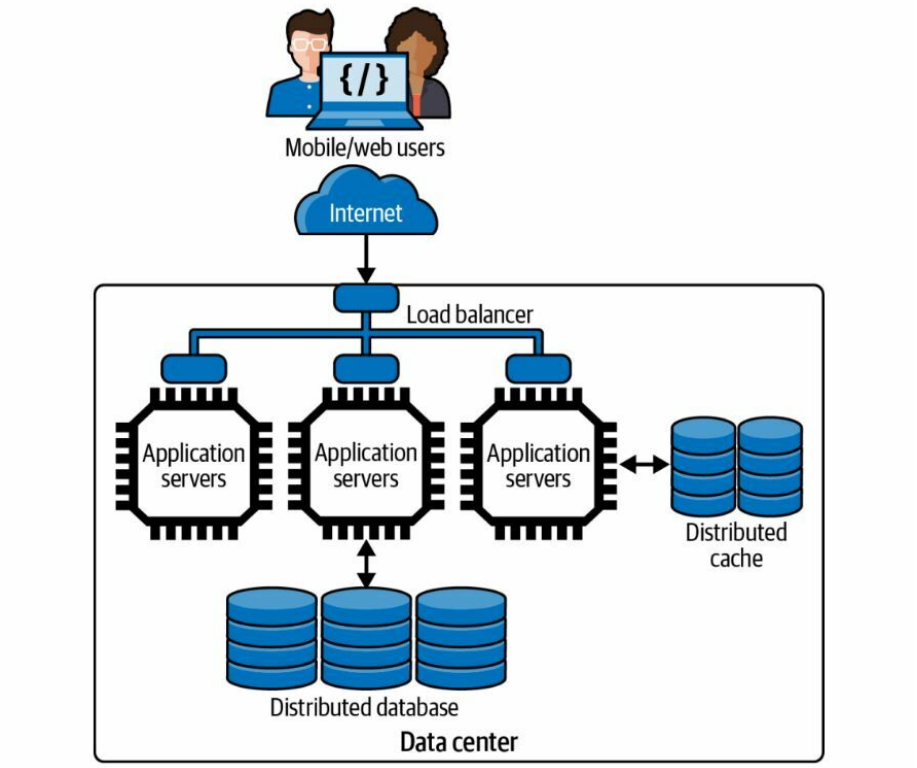


Рисунок 2.3 – Компоненти розподіленої архітектури

Розподілена архітектура має ряд переваг.

По-перше, це масштабованість: Систему можна масштабувати, додавши додаткові сервера, що збільшить обчислювальну потужність та продуктивність системи.

По-друге, це відмовостійкість: Розподіл операцій та ресурсів між безліччю серверів зменшує ризик повного збою системи.

По-третє, це паралельна обробка: Одні і ті ж самі операції може виконуватися кількома серверами одночасно, що прискорить виконання завдань.

І нарешті, загальний доступ: У розподіленій системі можуть спільно використовуватися дані, певне обладнання або програмне забезпечення [27].

Також у розподіленої архітектури є недоліки.

В першу чергу, це складність: Управління та узгодження безлічі серверів може бути досить складним завданням.

Друге – це висока залежність від мережної інфраструктури та потенційні затримки в роботі мережі.

Суттєвий недолік – це збільшення вартості експлуатації: Розподілені системи можуть вимагати досить великих початкових та експлуатаційних витрат.

І нарешті, складність виявлення помилок: Хоча виявлення збоїв спрощується, все одно діагностика та усунення помилок у розподіленій системі може бути досить складною [29].

Розподілені системи зазвичай використовуються у великих та складних системах, де потрібна досить висока масштабованість, доступність та відмовостійкість.

До прикладів успішно працюючих веб-додатків можна віднести належать такі великі корпоративні систами, як Google, Amazon, Facebook тощо.

Мікросервісна архітектура – це такий підхід до розробки програмного забезпечення, який включає в себе розділення програми на набір окремих

незалежних сервісів. При цьому кожен сервіс виконує свою певну функцію та обмінюється з іншими сервісами з допомогою різних механізмів, зазвичай HTTP API (рис. 2.4).

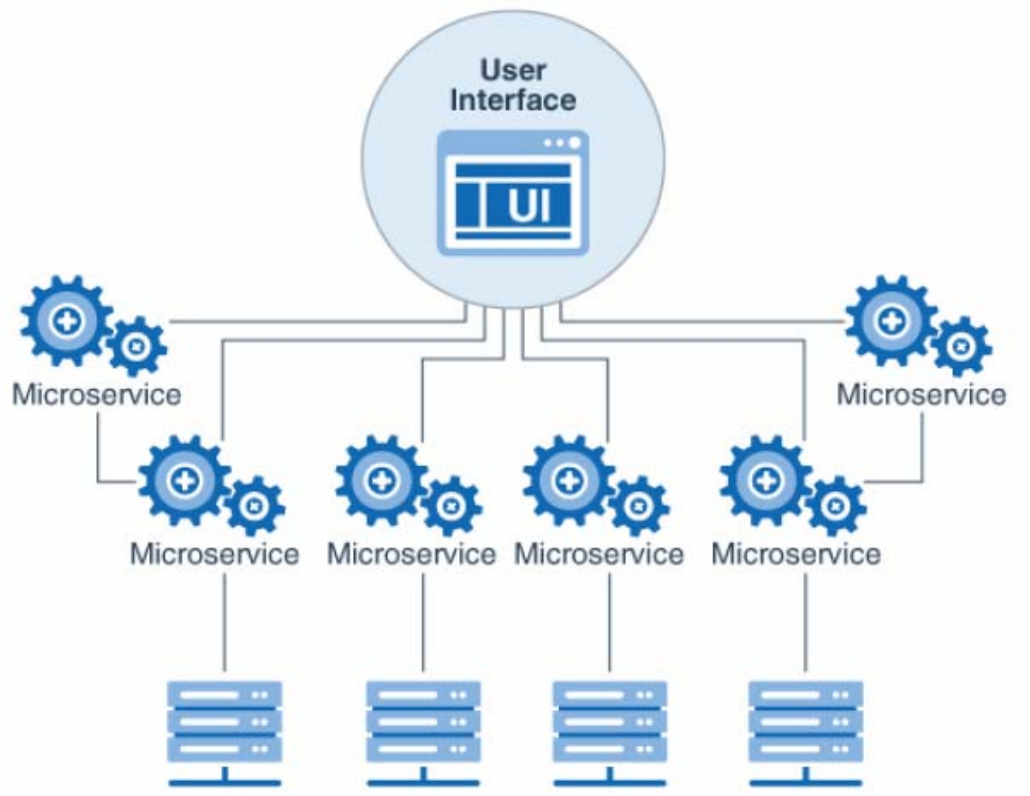


Рисунок 2.4 – Компоненти мікросервісної архітектури

В якості переваг мікросервісної архітектури можна назвати наступні:

Перше – це гнучкість у використанні технологій: Різні сервіси можуть використовувати різні технології та можуть бути написані на різних мовах програмування.

Наступне – це спрощення розробки та підтримки: Менший обсяг коду у кожного сервісу суттєво спрощує його розуміння та зміну.

Також важлива перевага – це масштабованість: Сервіси можна масштабувати окремо один від одного, що дозволяє якісно оптимізувати системні ресурси.

І також – це відмовостійкість: Збій в роботі одного сервісі практично не впливає на роботу всіх інших.

І нарешті – швидке розгортання нових функцій: Нові функції швидко впроваджуються та тестуються, оскільки не торкаються всього проекту в цілому [30].

Але мікросервісна архітектура також має недоліки.

По-перше, це складність управління: Велика кількість сервісів вимагає досить складної складної оркестрації та моніторингу їх роботи.

Також можуть виникнути проблеми з продуктивністю: Взаємодія сервісів між собою через мережу може спровокувати затримки.

І також важливий недолік - це складність тестування: Інтеграційне тестування взаємодії окремих сервісів значно складніше, ніж у монолітній системі [31].

Мікросервісна архітектура добре підходить для складних та великих систем, де потрібна масштабованість, гнучкість та швидке внесення змін. При цьому вона вимагає дуже ретельного планування та управління для гарантування продуктивності та стабільності системи.

В результаті проведеного аналізу ми обираємо монолітну архітектуру. Це обумовлено тим, що додаток має працювати у кожного провайдера окремо і не буде суттєво навантажений. Водночас, відмовлено від клієнт-серверної, розподіленої та мікросервісної архітектур через їхні недоліки, такі як складність розробки та розгортання, ускладнене управління додатком і підвищений ризик випадкових збоїв через розділений характер системи.

2.2 Вибір мови програмування

Програма веб-сервера забезпечує доступність інтернет-сторінок та інших онлайн-ресурсів. Ці сервери — це спеціалізовані комп'ютери, що мають безперервне з'єднання з мережею Інтернет.

Коли користувач запитує URL-адресу через браузер, останній надсилає запит до веб-сервера, який обробляє цей запит і надсилає відповідь у вигляді веб-сторінки. Ця сторінка складається з HTML, який браузер розшифровує,

відтворюючи текст, графіку, відео та інші компоненти на екрані користувача. Веб-сервер також може надавати доступ до інших онлайн-ресурсів, включаючи ігри, додатки та сервіси.

Розробка веб-сервера включає аналіз вимог, визначення основних характеристик, масштабованості, безпеки, а також проектування архітектури, кодування, тестування та впровадження на сервері.

Серверні скрипти зазвичай працюють з базами даних, такими як Oracle, MySQL, Microsoft SQL Server, MongoDB тощо. На відміну від клієнтських скриптів, які активуються в браузері користувача, серверні скрипти потребують обробки перед запуском і виконуються за допомогою спеціалізованого програмного забезпечення на сервері.

Серед технологій для створення серверних скриптів виділяються PHP, Node.js, Java, Python [32].

Java – це потужна та популярна мова програмування, яка широко використовується в розробці серверних додатків, включаючи монолітні архітектури.

Java є кросплатформною мовою, що означає, що програми, написані на Java, можуть бути запущені на різних операційних системах без змін.

Java поставляється з великим набором стандартних бібліотек, які полегшують розробку програм. Ці бібліотеки включають інструменти для роботи з мережею, базами даних, багатопоточністю і багатьма іншими завданнями.

Java добре масштабується як вертикально (шляхом збільшення ресурсів сервера), так і горизонтально (шляхом додавання додаткових серверів).

Java має безліч інтегрованих середовищ розробки (IDE), таких як Eclipse, IntelliJ IDEA та NetBeans, які надають розробникам потужні інструменти для написання, налагодження та тестування коду.

Java є строго типізованою мовою програмування, що забезпечує велику безпеку коду та зменшує кількість помилок часу виконання [33].

Переваги Java в монолітних архітектурах:

Простота розгортання: Програми Java легко розгортати та масштабувати в монолітній архітектурі, оскільки весь код знаходиться в одному додатку.

Широка підтримка: Java має велику спільноту розробників і багату документацію, що полегшує підтримку та супровід монолітних додатків цією мовою.

Сумісність з існуючими рішеннями: Багато існуючих корпоративних програм та інструментів розробки написані на Java, що забезпечує легку інтеграцію з ними при розробці монолітних додатків.

Недоліки Java у монолітних архітектурах:

Обмежена масштабованість: Монолітні програми можуть зіткнутися з обмеженнями масштабування через їх архітектурний підхід, що може призвести до складнощів в управлінні зростанням трафіку та навантаження.

Монолітність як єдиний варіант: У монолітній архітектурі немає гнучкості у виборі технологій та інструментів, оскільки весь код знаходиться в одному додатку. Це може обмежувати можливості використання нових технологій та підходів.

Повільне розгортання змін: При розробці монолітних програм зміни в одній частині програми можуть вимагати перескладання та перезапуску всієї програми, що може сповільнити процес розгортання.

Ускладнення підтримки зі зростанням розміру кодової бази: У міру зростання монолітної програми стає все складніше підтримувати та змінювати його кодову базу через її масштаб і складність [34].

Загалом Java надає широкий спектр інструментів і можливостей для розробки монолітних додатків, але також супроводжується деякими обмеженнями, пов'язаними з цим архітектурним підходом.

PHP – це скриптова мова програмування, яка часто використовується для розробки серверних програм та динамічних веб-сайтів.

PHP простий у освоєнні, тому що його синтаксис нагадує структуру інших широко використовуваних мов, таких як C та Perl. Це робить його привабливим вибором для розробників-початківців.

PHP був спеціально розроблений для роботи з веб-серверами, і він має багаті можливості для створення динамічних веб-сайтів та веб-додатків.

PHP має величезну спільноту розробників, що забезпечує доступ до безлічі бібліотек, фреймворків та рішень для різних завдань. Також є велика документація та посібники для початківців.

PHP надає розробникам широкий спектр можливостей для роботи з базами даних, файловою системою, мережею та іншими аспектами веб-розробки [35].

Переваги PHP у монолітних архітектурах:

Простота розгортання: Монолітні програми на PHP легко розгортати, так як вони зазвичай включають лише одну виконувану програму, яка може бути запущена на сервері.

Широке поширення та підтримка: PHP є однією з найпоширеніших мов програмування для веб-розробки, що забезпечує широку підтримку та доступ до ресурсів для розробників.

Багатий набір інструментів та фреймворків: PHP має безліч фреймворків та інструментів, таких як Laravel, Symfony, CodeIgniter та інші, які полегшують розробку монолітних програм, надаючи готові рішення для поширених завдань.

Недоліки PHP у монолітних архітектурах:

Складність підтримки при зростанні: При розвитку монолітної програми на PHP може виникнути складність підтримки та супроводу коду через його масштаб і складність.

Обмежена масштабованість: Монолітні програми на PHP можуть зіткнутися з обмеженнями масштабування через свою архітектуру, що може призвести до проблем з продуктивністю та керуванням навантаженням.

Однорідність технологій: У монолітній архітектурі на PHP іноді складно використовувати нові технології або інструменти, так як вся програма розробляється однією мовою і в рамках однієї технологічної платформи [36].

В цілому, PHP є потужним інструментом для розробки монолітних додатків завдяки своїй простоті, гнучкості та широкій підтримці, проте варто враховувати його обмеження при проектуванні та розгортанні великих додатків.

Node.js – це середовище виконання JavaScript, побудоване на движку V8 від Google Chrome, яке дозволяє запускати JavaScript на сервері.

Використання JavaScript як на стороні клієнта, так і на стороні сервера дозволяє розробникам створювати програми з однорідним кодом, що полегшує супровід та розвиток програми [37].

Node.js заснований на циклі подій та асинхронному програмуванні, що дозволяє обробляти велику кількість запитів без блокування потоку виконання, що особливо корисно для монолітних додатків з високим навантаженням.

Node.js має широкий набір бібліотек та модулів для різних завдань, таких як робота з мережею, файловою системою, базами даних та іншими аспектами веб-розробки.

Node.js має активну спільноту розробників, яка постійно створює нові інструменти, бібліотеки та фреймворки, розширюючи екосистему Node.js і роблячи її більш потужною та гнучкою [38].

Переваги Node.js у монолітних архітектурах:

Висока продуктивність: Завдяки асинхронному програмуванню Node.js забезпечує високу продуктивність при обробці великої кількості запитів, що робить його добрим вибором для монолітних програм з високим навантаженням.

Масштабованість: Node.js легко масштабується як вертикально (шляхом збільшення ресурсів сервера), так і горизонтально (шляхом

додавання додаткових серверів), що робить його придатним для монолітних програм зі зростанням трафіку.

Єдина мова: Використання однієї мови на клієнті та сервері спрощує розробку, налагодження та супровід програми, оскільки розробники можуть використовувати одні й ті самі інструменти та підходи на обох сторонах.

Недоліки Node.js у монолітних архітектурах:

Необхідність уважного керування ресурсами: Через асинхронну природу Node.js неправильне використання асинхронних операцій може призвести до витоків пам'яті та інших проблем з керуванням ресурсами.

Обмеження у продуктивності при обчислювальних операціях: Node.js не є оптимальним вибором для обчислювально-інтенсивних операцій через однопоточну модель виконання, що може призвести до блокування програми при виконанні довгих операцій.

Складність синхронізації та контролю залежностей: При розробці великих монолітних додатків на Node.js може виникнути складність із керуванням залежностями та синхронізацією асинхронних операцій, що потребує уважного проектування та структурування програми [39].

В цілому, Node.js надає потужні інструменти для розробки високопродуктивних та масштабованих монолітних додатків завдяки своїй асинхронній природі та єдиній мові на клієнті та сервері, але потребує уважного управління ресурсами та залежностями.

Python – це високорівнева мова програмування, яка широко використовується в різних сферах, включаючи веб-розробку.

Python відомий своєю простотою та виразністю, що робить його привабливим для розробників усіх рівнів. Синтаксис, що легко читається, спрощує супровід та розвиток додатків.

Python має широкий набір стандартних бібліотек, а також безліч сторонніх бібліотек і фреймворків, які полегшують розробку програм різного призначення, включаючи веб-розробку, обробку даних та машинне навчання.

Python доступний для багатьох платформ, включаючи Windows, macOS і різні дистрибутиви Linux, що робить його універсальним вибором для розробки монолітних програм на різних серверах.

Python легко інтегрується з іншими мовами програмування та технологіями, що дозволяє використовувати його в різних екосистемах та середовищах [40].

Переваги Python у монолітних архітектурах:

Простота розробки та супроводу: Завдяки своїй простоті та читання, Python полегшує розробку та супровід монолітних додатків, зменшуючи витрати на розробку та час на навчання нових розробників.

Велика спільнота та підтримка: Python має величезну спільноту розробників, що забезпечує доступ до безлічі ресурсів, бібліотек та фреймворків для розробки та супроводу монолітних програм.

Безліч інструментів та фреймворків: Python має багатий набір інструментів та фреймворків для розробки веб-додатків, таких як Django, Flask, Pyramid та інші, що спрощує розробку монолітних додатків цією мовою.

Недоліки Python у монолітних архітектурах:

Продуктивність: Python не завжди є найпродуктивнішою мовою програмування, особливо при виконанні обчислювально інтенсивних завдань, що може призвести до проблем з продуктивністю в монолітних програмах з високим навантаженням.

Обмежена підтримка асинхронного програмування: У порівнянні з деякими іншими мовами, такими як Node.js, Python має обмежену підтримку асинхронного програмування, що може ускладнити обробку великої кількості запитів у монолітних додатках з високим навантаженням.

Однопотокова модель виконання: За замовчуванням Python використовує глобальне блокування інтерпретатора (GIL), що обмежує його здатність ефективно використовувати багатопотоковість та масштабування на багатоядерних процесорах [41].

В цілому, Python надає потужні інструменти та можливості для розробки монолітних додатків завдяки своїй простоті, багатим бібліотекам та широкій підтримці, але потребує уважного обліку його обмежень при проектуванні та розгортанні великих додатків.

В результаті проведеного аналізу ми робимо вибір на користь PHP для розробки нашої монолітної архітектури, що можна обгрунтувати кількома ключовими аспектами:

1. Простота та швидкість розробки: PHP відомий своєю простотою та інтуїтивно зрозумілим синтаксисом, що робить його привабливим для швидкої розробки простих та середніх за складністю монолітних додатків. Завдяки широкому спектру інструментів та фреймворків, таких як Laravel, Symfony та CodeIgniter, PHP забезпечує гнучкість та прискорення процесу розробки.

2. Велика спільнота та екосистема: PHP має величезну спільноту розробників та велику екосистему бібліотек та фреймворків, що полегшує доступ до ресурсів, підтримки та вирішення проблем у процесі розробки монолітної програми.

3. Широке поширення та підтримка хостинг-провайдерів: PHP є однією з найпоширеніших мов програмування для веб-розробки та широко підтримується більшістю хостинг-провайдерів, що забезпечує легкість розгортання та масштабування монолітних програм на цій платформі.

4. Продуктивність: У контексті монолітної архітектури, де потрібна обробка великого обсягу запитів, PHP забезпечує хорошу продуктивність при правильній оптимізації та використанні кешування, особливо при роботі з традиційними веб-додатками.

5. Простота у підтримці та супроводі: PHP-код легко читаємо та зрозумілий, що полегшує його підтримку та супровід протягом життєвого циклу монолітного додатку. Низький поріг входження для нових розробників також забезпечує гнучкість у командній роботі та управлінні проектом.

2.3 Вибір системи управління базами даних

При визначенні типу бази даних, важливим вибором є рішення між SQL (реляційними) та NoSQL (нереляційними) системами. Обидві категорії пропонують життєздатні рішення, але мають ключові відмінності, які необхідно враховувати. Реляційні бази даних, відомі як RDBMS, традиційно використовують SQL для управління даними, в той час як NoSQL бази даних часто пов'язані з більш гнучкими і розподіленими структурами.

SQL бази даних використовують мову структурованих запитів, яка є потужним інструментом для обробки складних запитів. Однак, SQL вимагає від користувачів встановлення чіткої схеми даних до їх обробки, і всі дані повинні слідувати цій схемі, що може вимагати значної підготовки і ускладнювати зміни структури.

NoSQL бази даних пропонують динамічні схеми для неструктурованих даних, зберігаючи інформацію в різноманітних форматах, таких як документо-орієнтовані, стовпчасті, графові, або ключ-значення сховища. Це дозволяє створювати документи без визначення структури заздалегідь, і кожен документ може мати свою унікальну структуру, з можливістю додавання нових полів у процесі.

Масштабування баз даних також відрізняється: SQL бази даних зазвичай масштабуються вертикально, додаючи ресурси до одного сервера, тоді як NoSQL бази даних масштабуються горизонтально, розподіляючи навантаження між багатьма серверами.

Вертикальне масштабування може бути простішим у впровадженні та економічно ефективним, але має обмеження за ресурсами та може бути вартісним через необхідність модернізації серверів.

Горизонтальне масштабування пропонує майже необмежену масштабованість і надійність, але може бути складнішим у реалізації та вимагати додаткової інфраструктури, такої як балансування навантаження.

Структури баз даних SQL та NoSQL також різняться. SQL бази даних організовані у таблиці з рядками та стовпцями, де рядки представляють записи, а стовпці - атрибути. NoSQL бази даних можуть мати різні структури, включаючи пари ключ-значення, документи, графи та сховища з широкими стовпцями, кожна з яких підходить для певних типів даних і сценаріїв використання [42].

У веб-додатках переважно застосовуються SQL бази даних, що обґрунтовано декількома ключовими перевагами, які вони надають:

1. Структуроване зберігання даних: SQL бази даних організують дані у вигляді таблиць з чітко визначеними відносинами між ними, що забезпечує ефективне та структуроване зберігання даних. Це особливо важливо для веб-додатків, де часто потрібно зберігати інформацію про користувачів, замовлення, продукти та інші сутності.

2. Потужні можливості запитів SQL бази даних надають багатий набір операторів для виконання складних запитів до даних. Це дозволяє ефективно витягувати потрібну інформацію з бази даних для відображення на веб-сторінках або виконання різних операцій, таких як пошук, фільтрація та сортування.

3. Транзакційна безпека: SQL бази даних підтримують транзакції з використанням механізмів ACID (атомарність, узгодженість, ізольованість, довговічність), що забезпечує надійність та цілісність даних. Це особливо важливо для веб-застосунків, де необхідно забезпечити надійність при роботі з даними, таких як обробка замовлень або фінансові транзакції.

4. Масштабованість і продуктивність: SQL бази даних зазвичай мають хорошу продуктивність і масштабованість, особливо при правильному індексуванні та оптимізації запитів. Це дозволяє обробляти великі обсяги даних та високе навантаження веб-додатків.

5. Широка підтримка та поширеність: SQL бази даних, такі як MySQL, PostgreSQL, Microsoft SQL Server та інші, є поширеними і мають великі спільноти розробників та екосистеми інструментів. Це забезпечує легкість у

підтримці, розгортанні та супроводі веб-додатків, заснованих на SQL базах даних [43].

Що стосується вибору реляційної системи управління базами даних (SQL), то популярні два варіанти – PostgreSQL та MySQL. Обидві системи існують уже кілька десятиліть і зарекомендували себе як надійні, безпечні та масштабовані. Однак вони мають різні сильні та слабкі сторони, що робить одну з них більш придатною для конкретних випадків використання.

MySQL – це одна з найпопулярніших реляційних баз даних, що широко використовується веб-розробниками.

MySQL заснована на реляційній моделі даних, яка організує дані у вигляді таблиць з певними відносинами між ними. Це забезпечує структуроване та ефективне зберігання даних.

MySQL повністю сумісна з мовою SQL (Structured Query Language), що забезпечує багаті можливості запитів до даних, включаючи операції вибірки, вставки, оновлення та видалення.

MySQL має хорошу продуктивність і масштабованість, особливо при правильному індексуванні та оптимізації запитів. Він здатний обробляти великі обсяги даних і високе навантаження, що робить його сприятливим вибором для веб-додатків з високою продуктивністю.

MySQL є відкритим програмним забезпеченням та доступний для безкоштовного використання, що робить його доступним та привабливим для розробників та організацій з обмеженим бюджетом.

MySQL має величезну спільноту розробників та велику екосистему інструментів та бібліотек, що забезпечує легкість у підтримці, розгортанні та супроводі додатків, що використовують MySQL базу даних [44].

Переваги MySQL:

Простота використання та встановлення: MySQL легко встановлюється і налаштовується, що робить його привабливим для розробників-початківців та швидкого розгортання проектів.

Хороша продуктивність та масштабованість: MySQL забезпечує високу продуктивність та масштабованість при обробці великих обсягів даних та високому навантаженні.

Велика спільнота та підтримка: MySQL має величезну спільноту розробників та велику документацію, що забезпечує доступ до ресурсів та підтримки у разі виникнення проблем.

Надійність і стабільність: MySQL широко використовується в індустрії та має репутацію надійної та стабільної бази даних, що робить його хорошим вибором для критично важливих програм.

Недоліки MySQL:

Обмежена підтримка деяких типів даних: MySQL може мати обмеження підтримки деяких типів даних, особливо для зберігання напівструктурованих даних.

Складність горизонтального масштабування: Горизонтальне масштабування MySQL може бути складним і потребує додаткових зусиль та інструментів.

Можливі проблеми з продуктивністю при неправильному використанні: Неправильне індексування або оптимізація запитів може призвести до проблем з продуктивністю та навантаженням на базу даних [45].

Загалом, MySQL є потужним, надійним та поширеним рішенням для зберігання даних у веб-додатках, забезпечуючи високу продуктивність, простоту використання та широку підтримку спільноти.

PostgreSQL - це потужна об'єктно-реляційна система управління базами даних (ORDBMS), яка є одним із найрозвиненіших і розширюваних рішень у своєму класі.

PostgreSQL поєднує переваги реляційних та об'єктно-орієнтованих баз даних, дозволяючи зберігати та обробляти складні типи даних, такі як JSON, XML, масиви та типи даних користувача.

PostgreSQL повністю сумісна з мовою SQL (Structured Query Language) та надає багатий набір операторів для виконання різних запитів до даних, включаючи операції вибірки, вставки, оновлення та видалення.

PostgreSQL підтримує транзакції з використанням механізмів ACID (атомарність, узгодженість, ізольованість, довговічність), що забезпечує надійність та цілісність даних.

PostgreSQL має хорошу продуктивність і масштабованість, особливо при правильному індексуванні та оптимізації запитів. Він здатний обробляти великі обсяги даних та високе навантаження.

PostgreSQL є відкритим програмним забезпеченням та доступний для безкоштовного використання, що робить його привабливим для розробників та організацій з обмеженим бюджетом.

PostgreSQL має розширювану архітектуру, яка дозволяє розробникам створювати та інтегрувати власні розширення та користувальницькі функції для задоволення специфічних потреб проекту [46].

Переваги PostgreSQL:

Повна підтримка SQL та стандартів: PostgreSQL повністю сумісний із SQL та підтримує безліч стандартів, що забезпечує сумісність та переносимість додатків між різними платформами.

Багаті можливості даних: PostgreSQL надає широкий набір функцій та можливостей для роботи з даними, включаючи підтримку різних типів даних, геоданих, повнотекстового пошуку та інших розширень.

Надійність та цілісність даних: PostgreSQL забезпечує високу надійність та цілісність даних завдяки підтримці ACID-транзакцій та механізмів забезпечення цілісності даних.

Активна спільнота та підтримка: PostgreSQL має активну спільноту розробників та велику документацію, що забезпечує доступ до ресурсів та підтримки у разі виникнення проблем.

Недоліки PostgreSQL:

Деяке обмеження у продуктивності: У порівнянні з деякими іншими СУБД, PostgreSQL може бути дещо повільнішим у деяких випадках через його більш складну архітектуру та високу надійність.

Складність адміністрування: Деякі аспекти адміністрування PostgreSQL можуть бути складними для нових користувачів через його багатий набір функцій та параметрів конфігурації.

Обмежена підтримка деяких розширень: Незважаючи на багатий набір функцій, деякі розширення та функціональні можливості можуть бути обмежені у PostgreSQL у порівнянні з іншими СУБД [47].

В цілому, PostgreSQL є потужним і гнучким рішенням для управління даними у веб-додатках, забезпечуючи високу надійність, продуктивність і розширюваність, хоча деякі аспекти його використання можуть вимагати додаткових зусиль в адмініструванні та налаштуванні.

В результаті проведеного аналізу ми робимо вибір на користь застосування MySQL баз даних у нашому веб-додатку, що обґрунтовано наступними ключовими перевагами:

1. Простота використання та налаштування: MySQL є однією з найпопулярніших баз даних, яка має простий і зрозумілий інтерфейс. Її легко встановлювати, налаштовувати та використовувати, що робить її привабливим вибором для веб-застосунків.

2. Широка підтримка та поширеність: MySQL широко використовується в індустрії та має широке співтовариство розробників та екосистему інструментів. Це забезпечує легкість у підтримці, розгортанні та супроводі веб-додатків, заснованих на MySQL базі даних.

3. Висока продуктивність: MySQL має хорошу продуктивність і масштабованість, особливо при правильному налаштуванні та оптимізації. Вона здатна обробляти великі обсяги даних та високе навантаження, що робить її підходящим вибором для веб-додатків з високою продуктивністю.

4. Надійність і стабільність: MySQL є стабільною та надійною базою даних, що широко використовується в критично важливих додатках. Вона

забезпечує підтримку ACID-транзакцій та механізмів забезпечення цілісності даних, що забезпечує надійність даних у веб-додатках.

5. Безкоштовність та відкритий вихідний код: MySQL доступний для безкоштовного використання та є відкритим програмним забезпеченням, що робить його доступним для широкого кола розробників та організацій.

6. Хороша інтеграція з іншими технологіями: MySQL легко інтегрується з іншими технологіями та мовами програмування, такими як PHP, Python, Ruby та іншими, що робить його зручним вибором для веб-застосунків на різних платформах [48].

2.4 Вибір технологій для побудови інтерфейсу користувача

Інтерфейс користувача є ключовим елементом системи, що забезпечує інтерактивність з кінцевим користувачем. Його основні завдання - це презентація даних та обробка введення з боку користувача.

Створення інтерфейсу користувача включає декілька фаз: від аналітичного вивчення вимог, які включають функціональність, нефункціональні характеристики та технічні специфікації, до дизайну, де формується зовнішній вигляд і структура інтерфейсу. Після цього наступає кодування, тестування для перевірки відповідності встановленим вимогам та запуск в роботу.

Існує безліч методик розробки інтерфейсу користувача, які можна розділити за різними параметрами, такими як метод доставки до користувача (завантаження, вбудовані системи), вид інтерфейсу (текстовий, візуальний) та використовувані технології (HTML, CSS, JavaScript, Java, .NET, конструкторські платформи).

Важливо обирати методику розробки, виходячи з унікальних потреб системи, її цілей, бюджету, часових рамок, а також знань та досвіду команди розробників. Наприклад, самостійна розробка інтерфейсу користувача дає повний контроль над проектом, але може бути складною і дорогою. З іншого

боку, використання готових конструкторів може значно прискорити процес, але зменшити можливості для індивідуалізації. Вибір залежить від специфіки проекту та доступних ресурсів [49].

У даному проекті віддається перевага ручному кодуванню інтерфейсу без застосування готових конструкторів, оскільки розробник має достатній досвід для подолання можливих труднощів, що дозволяє зберегти контроль над всіма аспектами розробки.

HTML (Hypertext Markup Language) - це код, який використовується для структурування та відображення веб-сторінки та її контенту.

Фактично, HTML не є мовою програмування – це мова розмітки, і використовується, щоб повідомляти браузері, як відображати веб-сторінки, які він відвідує. Розмітка може бути складною або простою, залежно від того, як хоче веб-дизайнер. HTML складається з ряду елементів, які використовуються, щоб вкладати або обертати різні частини контенту, щоб змусити контент відображатись або діяти певним чином.

Мова розмітки – одна з найпростіших з погляду програмування. Знаючи теги, можна без проблем написати будь-яку веб-сторінку. Однак використовуючи виключно цю мову, дизайнер отримає стандартний стиль, що практично ніде сьогодні не зустрічається, окрім дуже стародавніх сайтів, дизайн яких залишився незмінним з моменту їх появи [50].

На відміну від HTML, CSS – це мова стилів. Вона не є самостійною, тобто без HTML її існування марне. Коли створюється веб-сторінка HTML, стилі CSS додаються окремим файлом і підключаються до сторінки HTML (хоча можливі варіанти, за яких деякі стилі прописуються прямо в HTML тегах, але найкраща практика – писати стилі в окремому документі та підключати його до HTML).

Що вміє CSS? Наприклад, стандартний HTML-тег <button></button> це звичайна сіра кнопка. За допомогою CSS можна не тільки змінити її колір та розмір, але й, наприклад, зробити невелику анімацію.

Незважаючи на те, що розробка CSS розпочалася ще на початку 90-х років минулого століття, перший офіційний документ з цієї мови був опублікований лише у 1996 році. У першій рекомендації пропонувалися широкі можливості форматування тексту. Наприклад, можна було змінювати параметри шрифту, додавати кольори, різні атрибути, а також додавати властивості блоків.

Поступово мова стилів змінювався на краще. Додався новий функціонал, який дозволяв працювати вже не лише з текстом. Але і з рештою елементів на сторінці.

Враховуючи те, що практично будь-який сайт в інтернеті написаний за допомогою HTML, говорити про переваги цієї мови розмітки досить складно. Адже без неї цих сайтів просто не було б. Тому розглянемо основні переваги саме CSS:

1. Відносна простота використання. Освоїти CSS можна досить швидко. Принаймні, на це піде менше часу, ніж вивчення мов програмування на кшталт JavaScript. Всі стилі прописуються в одному файлі і їх можна використовувати для всіх сторінок файлу або веб-додатку.

2. Зменшення розміру веб-сайту. CSS прописується в окремому файлі, а потім підключається до HTML документа. Саме завдяки цьому вдається скоротити розмір HTML-сторінки. Після завантаження сайту браузером CSS-файл кешується. Відповідно, стилі надалі будуть використані для всіх сторінок. Завантажувати їх знову не потрібно.

3. Безліч додаткових можливостей щодо стилізації тексту та іншого контенту. За допомогою CSS, наприклад, можна зробити обтікання тексту або кнопки іншим текстом.

4. Немає необхідності робити структуру макета табличною. До появи CSS макети сторінок робилися у вигляді таблиць, що дозволяло легко позиціонувати будь-який елемент у потрібному місці. Однак це уповільнювало швидкість завантаження. До того ж, код виходив громіздким

та незручним. Можливість роботи з контейнерами (div) із появою CSS вирішила цю проблему.

5. Постійне оновлення. CSS, як і HTML постійно оновлюються. У нові версії додається корисний функціонал, який дозволяє не лише спростити роботу, а й розширити можливості цих двох мов.

Що стосується недоліків, у CSS він лише один - контент по-різному відображається в різних браузерах. У застарілих браузерах підтримуються в повному обсязі функції сучасного CSS [51].

При виборі технологій розробки клієнтської частини системи буде доцільно також провести аналіз основних бібліотек та фреймворків: React, Vue та Angular. Цей аналіз дозволить обґрунтувати вибір конкретних технологій для забезпечення оптимальної функціональності та ефективності системи.

React — це бібліотека JavaScript, розроблена Facebook для створення інтерфейсів користувача. Вона особливо популярна створення односторінкових додатків (SPA).

Основні характеристики React:

1. Компонентний підхід: Програми React складаються з незалежних компонентів, кожен з яких управляє своїм власним станом і логікою відображення. Компоненти можуть бути функціональними чи класовими.

2. JSX: React використовує JSX (JavaScript XML), синтаксис розширення, який дозволяє писати HTML-подібний код у JavaScript. JSX полегшує створення та читання компонентів.

3. Віртуальний DOM: React підтримує віртуальний DOM, який дозволяє оптимізувати оновлення інтерфейсу користувача, зменшуючи кількість реальних змін у DOM.

4. Односпрямований потік даних: У React дані передаються від батьківських компонентів до дочірнім через властивості (props), що робить дані та стан більш передбачуваними.

5. Hooks: React hooks, такі як useState і useEffect, дозволяють функціональним компонентам керувати станом і побічними ефектами, що робить код компактнішим і зрозумілішим.

Vue - це прогресивний фреймворк JavaScript, розроблений для створення інтерфейсів користувача. Він був створений Еваном Ю і націлений на легкість та інтеграцію.

Основні характеристики Vue:

1. Реактивні дані: Vue використовує реактивну систему, яка відстежує зміни в даних і автоматично оновлює інтерфейс користувача.

2. Шаблони: Vue використовує шаблони, які дозволяють писати HTML-подібний код з використанням директив та прив'язок даних для динамічного рендерингу.

3. Компонентна архітектура: Подібно до React, Vue будується на компонентах, які інкапсулюють HTML, CSS і JavaScript.

4. Директиви: Vue надає директиви, такі як v-bind, v-model, v-for та v-if, які полегшують створення інтерактивних інтерфейсів.

5. Інтеграція та масштабованість: Vue можна легко інтегрувати в існуючі проекти та масштабувати від невеликих до великих програм.

Angular – це фреймворк JavaScript (та TypeScript), розроблений Google. Він призначений для створення динамічних та складних веб-додатків.

Основні характеристики Angular:

1. TypeScript: Angular написаний на TypeScript, що забезпечує строгу типізацію та покращену підтримку IDE.

2. Модульна архітектура: Angular використовує модульну архітектуру, де програми діляться на модулі для кращої організації та повторного використання коду.

3. Декларативні шаблони: Angular використовує декларативні шаблони, які включають прив'язку даних, директиви та фільтри.

4. Dependency Injection: Angular має вбудовану систему Dependency Injection (DI), яка спрощує керування залежностями та сприяє модульності.

5. RxJS: Angular активно використовує RxJS для обробки асинхронних даних через реактивні потоки [52].

Переваги використання фреймворків:

1. Прискорення розробки: Фреймворки надають безліч готових компонентів та рішень, що дозволяє швидше розробляти функціонал.

2. Підтримка та спільнота: Популярні фреймворки мають великі спільноти, документацію та ресурси, що полегшує навчання та вирішення проблем.

3. Структура та масштабованість: Фреймворки допомагають організувати код і зробити його більш підтримуваним та масштабованим, особливо у великих проектах.

Недоліки використання фреймворків:

1. Складність та надмірність: Для простих проектів використання фреймворків може призвести до надмірної складності та ускладнити кодову базу.

2. Продуктивність: Фреймворки додають додаткову вагу до проекту, що може негативно позначитися на продуктивності, особливо якщо проект невеликий.

3. Витрати часу: Застосування фреймворку вимагає часу та зусиль. Для простих проектів може бути недоцільно витрачати ресурси, коли можна обійтися чистим HTML, CSS і JavaScript.

4. Залежність від фреймворку: Використання фреймворку створює залежність від нього, що може бути проблемою при необхідності зміни технологій або якщо фреймворк перестає підтримуватись [53].

Таким чином, для простих проектів часто краще обходитися без використання фреймворків, оскільки це дозволяє зберегти простоту, легкість та продуктивність коду. Чистого HTML, CSS та JavaScript цілком достатньо для створення невеликих та простих веб-додатків.

2.5 Висновки до розділу 2

В результаті проведеного аналізу ми обрали монолітну архітектуру. Це обумовлено тим, що додаток має працювати у кожного провайдера окремо і не буде суттєво навантажений. Водночас, відмовилися від клієнт-серверної, розподіленої та мікросервісної архітектур через їхні недоліки, такі як складність розробки та розгортання, ускладнене управління додатком і підвищений ризик випадкових збоїв через розділений характер системи.

Також по результатах проведеного аналізу ми обрали PHP для розробки монолітної архітектури, що обумовлено його простотою, продуктивністю, широкою підтримкою спільноти та індустрії, а також наявністю багатої екосистеми інструментів та фреймворків, що робить його привабливим та ефективним вибором для створення нашого веб-додатку.

Вибір між SQL та NoSQL базами даних залежить від конкретних вимог проекту. SQL бази даних зазвичай краще підходять для додатків з жорсткою структурою даних, що вимагають надійності та ACID-сумісності, у той час як NoSQL бази даних частіше використовуються для додатків з великими обсягами даних, де важлива гнучкість та масштабованість. Загалом застосування SQL баз даних у веб-додатку забезпечує надійне, структуроване та ефективне зберігання даних, що є ключовим компонентом для успішного функціонування додатку та забезпечення задоволення потреб користувачів.

Серед SQL баз даних ми обрали застосування MySQL баз даних у розробці, що має забезпечити надійне, продуктивне та легко масштабоване сховище даних для програм будь-якого масштабу. Її простота використання, широка підтримка та висока продуктивність зробили її популярним та привабливим вибором для нашого веб-додатку.

У даному проекті віддається перевага ручному кодуванню інтерфейсу користувача без застосування готових конструкторів, оскільки розробник має достатній досвід для подолання можливих труднощів, що дозволяє зберегти контроль над всіма аспектами розробки.

Також аналіз продемонстрував, що для простих проектів часто краще обходитися без використання фреймворків, оскільки це дозволяє зберегти простоту, легкість та продуктивність коду. Чистого HTML, CSS та JavaScript цілком достатньо для створення невеликих та простих веб-додатків. Тому при розробці інтерфейсу в даному проекті буде віддана перевага застосування саме чистого HTML, CSS та JavaScript.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ АВТОМАТИЗАЦІЇ ПРОЦЕСІВ ПРОВАЙДЕРА БЕЗПЕРЕРВНОГО ПРОФЕСІЙНОГО РОЗВИТКУ МЕДИЧНИХ ПРАЦІВНИКІВ

3.1 Опис процесів в системі

У сфері створення програмного забезпечення особлива увага приділяється розробці та утриманню ефективних, високоякісних та стабільних програмних продуктів. Цей процес інтегрує техніки, методи та підходи з різноманітних дисциплін, включаючи комп'ютерні науки, управління проектами, математику, інженерію тощо. Розробка програмного забезпечення, як і інші технічні спеціальності, звертає увагу на аспекти якості, економічності та довговічності. Відомо, що деякі програмні продукти містять мільйони ліній коду, які мають бездоганно працювати в різних умовах.

Розроблювана система автоматизації процесів провайдера безперервного професійного розвитку медичних працівників включає дві основні ключові ролі: користувача та адміністратора. Кожна з цих ролей відповідає за конкретні процеси, які можуть бути або спільними, або унікальними для кожної з них.

З метою детального аналізу всіх ключових ролей учасників та виконання ними відповідних процесів розроблено Діаграму випадків використання (рис. 3.1).

Діаграма випадків використання (англ. Use case diagram) - це візуальний елемент мови моделювання UML, призначений для опису функцій, які система має надавати, та взаємодії з кінцевими користувачами. Цей інструмент демонструє, як користувачі взаємодіють із системою та які дії вона має здійснювати.

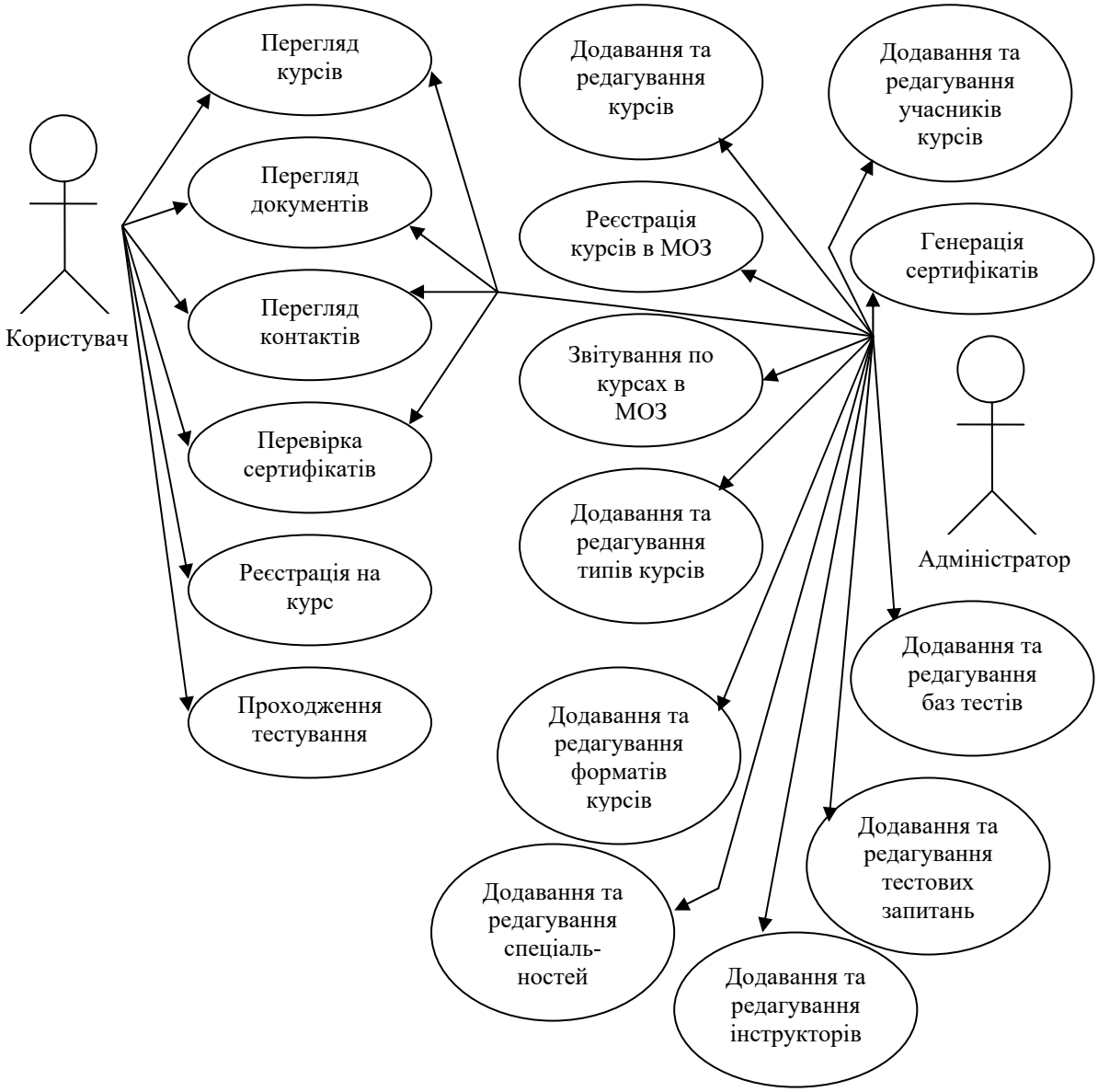


Рисунок 3.1 – Діаграма випадків використання

Дії Користувача не потребують авторизації. Всі доступні йому елементи керування розташовані на головній сторінці. Користувач має можливість виконувати наступні дії:

1. Переглядати список майбутніх курсів.

На даній сторінці динамічно відображаються всі актуальні навчальні заходи провайдера. Є можливість переглянути детальну інформацію про курс натиснувши на кнопку «Деталі» та подати заявку на реєстрацію натиснувши на кнопку «Реєстрація».

2. Переглядати список документів провайдера.

Дана сторінка є статичною, на ній розміщені документи провайдера, які вимагає центр тестування МОЗ:

- Витяг із Єдиного державного реєстру юридичних осіб, фізичних осіб – підприємців та громадських формувань із переліком КВЕД.

- Положення про оцінку заходів безперервного професійного розвитку на ознаки академічної доброчесності та дотримання принципів практичної медичної/фармацевтичної/реабілітаційної діяльності, заснованої на доказах, затверджене Провайдером.

- Методологія оцінювання набутих знань, компетентностей та практичних навичок працівників сфери охорони здоров'я, що затверджується Провайдером.

- Положення про запобігання конфлікту інтересів під час проведення заходів безперервного професійного розвитку та недопущення залучення і використання коштів фізичних (юридичних) осіб для реклами лікарських засобів, медичних виробів, допоміжних засобів реабілітації або медичних послуг, затверджене Провайдером.

Система дозволяє переглянути вказані документи.

3. Переглядати контактну інформацію.

Дана сторінка є статичною, на ній розміщена контактна інформація провайдера.

4. Перевіряти відповідність виданого сертифікату.

На даній сторінці Користувач має можливість перевірити дійсність сертифікату ввівши його номер та натиснувши кнопку «Перевірити». Якщо система знаходить відповідний сертифікат в базі даних, то відображає його дані – Назву курсу, Дату проведення, ПІБ учасника та інше, чим підтверджує факт проходження учасником відповідного курсу даного провайдера. Якщо сертифікат не знайдено – виводиться відповідне повідомлення.

5. Реєструватися на курс.

На даній сторінці Користувач має можливість подати заявку на реєстрацію на курс. Система запропонує Користувачу форму в якій

необхідно заповнити наступні обов’язкові дані, які вимагає центр тестування МОЗ:

- Прізвище учасника
- Ім'я учасника
- По батькові учасника (за наявності)
- Дата народження
- Електронна адреса
- Освіта
- Місце роботи
- Займана посада

Після заповнення всіх необхідних даних та натискання на кнопку «Відправити» система реєструє заявку учасника та видає відповідне підтвердження.

6. Проходження тестування.

На даній сторінці кожному учаснику буде потрібно вибрати себе зі списку та натиснути кнопку «Вибрати».

Система перенаправить учасника на проходження тестування де він має відповісти на питання, згідно налаштувань прописаних для курсу.

Для відповіді на запитання необхідно відмітити вірну, на думку учасника, відповідь та натиснути кнопку «Далі».

Якщо в налаштуваннях передбачено ліміт часу на відповідь на питання, то в формі тестування відображається таймер часу, що залишився. Після закінчення часу, система автоматично відмічає, що відповідь не була дана вчасно та переходить на наступне запитання.

Якщо в налаштуваннях передбачено пропуск запитань, то учасник має можливість натиснути кнопку «Пропустити» і перейти до наступного питання без фіксації відповіді. Питання буде задано знову після проходження останнього питання. Якщо був передбачений ліміт часу на відповідь на питання, то при пропуску питання система зберігає стан таймеру і відновлює його після повернення до питання.

Після проходження останнього запитання система виводить результати тестування в кількісному (кількість вірних відповідей з загальної кількості), відсотковому (відсоток вірних відповідей) та логічному вираженні (здав / не здав). Також виводиться перелік всіх питань та відповідей, які дав учасник для роботи над помилками.

Система також підраховує кількість виконаних спроб і якщо кількість спроб досягла ліміту, передбаченого в налаштуваннях, то виводить повідомлення про це та не дозволяє розпочати нову спробу.

У Адміністратора, після успішної авторизації в системі, з лівої сторони відображається меню дій, які йому доступні.

Адміністратор має можливість виконувати наступні дії:

1. Додавання та редагування курсів

Дана сторінка дозволяє створювати в системі нові курси шляхом натискання кнопки «Додати». У відповідь система відкриває форму, в яку потрібно внести наступні дані:

- Назва курса.
- Місто та адреса проведення курсу.
- Дати початку та завершення курсу.
- Час початку та завершення курсу.
- Мета проведення курсу.
- Методи проведення курсу.
- Методи контролю знань.
- Погодинна програма курсу.
- Опис структури курсу.
- Кількість балів БПР, які будуть нараховані учасникам.
- Тип курсу (перелік доступних типів курсів задається на відповідній сторінці).
- Формат проведення курсу (перелік доступних форматів проведення курсів задається на відповідній сторінці).
- Запланована кількість учасників.

- Ціна курсу.

- Тривалість курсу (в годинах).

- База тестових запитань, які будуть використані для контролю знань даного курсу (перелік доступних баз тестів задається на відповідній сторінці).

- Опції тестування – кількість питань, обмеження часу на одне запитання тощо.

- Медичні / фармацевтичні спеціальності, для яких буде актуальним даний курс (перелік доступних спеціальностей задається на відповідній сторінці).

- Інструктор або декілька інструкторів, які мають проводити курс (перелік доступних інструкторів задається на відповідній сторінці).

Після заповнення всіх необхідних даних та натискання кнопки «Зберегти» система вносить курс в базу даних.

Також система передбачає редагування даних курсу у випадку виявлених помилок. Для цього необхідно натиснути кнопку «Редагувати» біля відповідного курсу, внести корективи та натиснути кнопку «Зберегти».

Також передбачено видалити курс натиснувши кнопку «Видалити» біля відповідного курсу. Видалений курс можна відновити натиснувши кнопку «Відновити».

2. Реєстрація курсів в МОЗ

Дана сторінка дозволяє сформувати набір даних, який необхідний для реєстрації курсу в МОЗ:

- Реєстраційний номер провайдера.

- Тема курсу.

- Тип курсу.

- Формат курсу.

- Акредитація курсу («всеукраїнський», «акредитований за кордоном чи в Україні EACCME/ACCME/RCPSC чи сертифікований

ERC/ILCOR/AHA» або «курс відбувається в країні з високим рівнем доходу (за рейтингом Світового банку)»).

- Кількість балів БПР, що нараховуються відповідно до законодавства.
- Дата початку проведення курсу.
- Дата завершення проведення курсу.
- Цільова аудиторія (Медичні / Фармацевтичні спеціальності).
- Мова проведення курсу.
- Місце проведення.
- Посилання для реєстрації на курс.
- Посилання на веб-сайт провайдера, де розміщена інформація про курс.
- ПІБ контактної особи провайдера, яка відповідальна за організацію та проведення курсу.
- Номер телефону контактної особи провайдера, яка відповідальна за організацію та проведення курсу.

Також сторінка готує документи, які мають бути завантажені у форматі doc або docx - Програму та Навчальну програму заходу.

Програма курсу містить наступні дані:

- Вид курсу
- Тема курсу
- Дата
- Час початку
- Опис структури курсу
- Час завершення
- ПІБ Інструктора/Викладача/Тренера
- Резюме Інструктора/Викладача/Тренера
- Кількість інструкторів
- Кількість учасників

Навчальна програма курсу містить наступні дані:

- Тема курсу

- Вид курсу

- Цільова аудиторія (Медичні / Фармацевтичні спеціальності)

- Мета курсу

- Перелік компетентностей, що набуваються або вдосконалюються

- Опис структури курсу

- Загальний обсяг навчального навантаження (в годинах)

- Форми та методи організації та проведення курсу

- Матеріально-технічне забезпечення курсу

- Форми підсумкового контролю

Дані документи система дозволяє скачати.

3. Звітування по курсах в МОЗ

Дана сторінка дозволяє сформувати набір даних, який необхідний для звітування за проведення курсу перед МОЗ:

- Реєстраційний номер провайдера

- Номер курсу

- ПІБ контактної особи

- № телефону (мобільний) контактної особи

Також сторінка готує документи, які мають бути завантажені:

- Програма заходу у форматі doc або docx.

- Копія одного сертифіката заходу (оформленого повністю) у форматі pdf або зображення.

- Файл "Шаблон Видані сертифікати 2024" у форматі xls або xlsx.

- Файл "Контроль успішності курсантів" у форматі xls або xlsx.

Файл "Шаблон Видані сертифікати 2024" містить наступні дані:

- Реєстраційний номер Провайдера

- Реєстраційний номер курсу

- Номер сертифіката

- Прізвище, власне ім'я, по батькові (за наявності) учасника

- Нараховані бали БПР

- Дата народження учасника

- Електронна адреса учасника
- Освіта учасника
- Місце роботи учасника
- Посада учасника
- Результати оцінювання за проходження курсу

Файл "Контроль успішності курсантів" містить наступні дані:

- Унікальний номер тестування в системі
- Дата та час тестування
- ПІБ курсанта
- Результати тестування

4. Додавання та редагування типів курсів

Дана сторінка дозволяє створювати в системі нові типи курсів шляхом натискання кнопки «Додати». Після внесення назви нового типу курсу та натискання кнопки «Зберегти» система вносить інформацію в базу даних.

Також система передбачає редагування створених типів курсу у випадку виявлених помилок. Для цього необхідно натиснути кнопку «Редагувати» біля відповідного типу курсу, внести корективи та натиснути кнопку «Зберегти».

Також передбачено видалити тип курсу натиснувши кнопку «Видалити» біля відповідного типу. Видалений тип курсу можна відновити натиснувши кнопку «Відновити».

5. Додавання та редагування форматів курсів

Дана сторінка дозволяє створювати в системі нові формати курсів шляхом натискання кнопки «Додати». Після внесення назви нового формату курсу та натискання кнопки «Зберегти» система вносить інформацію в базу даних.

Також система передбачає редагування створених форматів курсу у випадку виявлених помилок. Для цього необхідно натиснути кнопку «Редагувати» біля відповідного формату курсу, внести корективи та натиснути кнопку «Зберегти».

Також передбачено видалити формат курсу натиснувши кнопку «Видалити» біля відповідного типу. Видалений формат курсу можна відновити натиснувши кнопку «Відновити».

6. Додавання та редагування спеціальностей

Дана сторінка дозволяє вносити в систему нові спеціальності шляхом натискання кнопки «Додати». Після внесення назви нової спеціальності та натискання кнопки «Зберегти» система вносить інформацію в базу даних.

Також система передбачає редагування створених спеціальностей у випадку виявлених помилок. Для цього необхідно натиснути кнопку «Редагувати» біля відповідної спеціальності, внести корективи та натиснути кнопку «Зберегти».

Також передбачено видалити спеціальність натиснувши кнопку «Видалити» біля відповідної спеціальності. Видалену спеціальність можна відновити натиснувши кнопку «Відновити».

7. Додавання та редагування учасників курсів

Дана сторінка дозволяє вводити в систему учасників курсу шляхом натискання кнопки «Додати».

У відповідь система відкриває форму, в яку потрібно внести наступні дані:

- Прізвище учасника
- Ім’я учасника
- По батькові учасника (за наявності)
- Стать учасника
- Дата народження учасника
- Електронна пошта учасника
- Освіта учасника
- Місце роботи учасника
- Посада учасника
- Курс, на який вводиться учасник (перелік доступних курсів задається на відповідній сторінці).

Після заповнення всіх необхідних даних та натискання кнопки «Зберегти» система вносить учасника в базу даних. Також система передбачає редагування введених учасників курсу у випадку виявлених помилок. Для цього необхідно натиснути кнопку «Редагувати» біля відповідного учасника, внести корективи та натиснути кнопку «Зберегти».

Також передбачено видалити учасника курсу натиснувши кнопку «Видалити» біля відповідного учасника. Видаленого учасника можна відновити натиснувши кнопку «Відновити».

8. Генерація сертифікату

Після натискання на дану кнопку біля відповідного учасника система генерує сертифікат в форматі pdf згідно заздалегідь підготовленого шаблону.

Сертифікат обов’язково містить наступну інформацію:

- Повну юридичну назву Провайдера (відповідно до Єдиного державного реєстру юридичних осіб, фізичних осіб – підприємців та громадських формувань).

- Вид курсу.

- Тему курсу.

- Прізвище, власне ім'я, по батькові (за наявності) отримувача сертифіката.

- Спеціальності за якими проводилось навчання (цільова аудиторія)

- Кількість балів БПР.

- Дата проведення курсу.

- Номер сертифікату.

- Підпис та ПІБ керівника Провайдера, який гарантує проведення оцінювання набутих знань та практичних навичок, а також дотримання всіх вимог законодавства під час проведення заходу.

Обов’язковим компонентом сертифікату є номер, який виражається у цифрах у наступному форматі: xxxx-zxxx-zxxxxxx-zxxxxx де,

X – будь-яка цифра

Z – будь-яка цифра, окрім 0 (рис. 3.2).

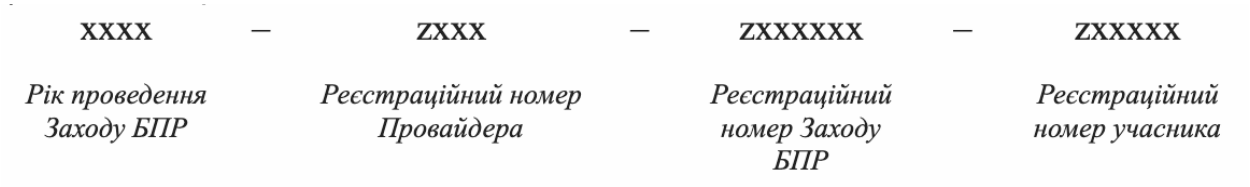


Рисунок 3.2 – Формат номеру сертифікату

Особливістю даного номеру є його унікальність, тобто ні в якому разі не має бути 2-х сертифікатів з однаковими номерами.

Рік проведення заходу БПР – це 4-х цифровий запис року і є однаковим для всіх курсів у відповідному році.

Реєстраційний номер Провайдера присвоюється МОЗ, ніколи не змінюється і є однаковим для всіх курсів відповідного Провайдера.

Реєстраційний номер курсу також присвоюється МОЗ ніколи не змінюється і є однаковим для всіх сертифікатів відповідного курсу.

Таким чином, унікальність номера забезпечується за рахунок Реєстраційного номера учасника, тому в системі передбачена генерація унікального номера учасника в рамках курсу.

9. Додавання та редагування баз тестів

Дана сторінка дозволяє створювати в системі нові бази тестів шляхом натискання кнопки «Додати». Після внесення назви бази та натискання кнопки «Зберегти» система вносить інформацію в базу даних.

Також система передбачає редагування створених баз тестів у випадку виявлених помилок. Для цього необхідно натиснути кнопку «Редагувати» біля відповідної бази, внести корективи та натиснути кнопку «Зберегти».

Також передбачено видалити базу тестів натиснувши кнопку «Видалити» біля відповідної бази. Видалену базу тестів можна відновити натиснувши кнопку «Відновити».

10. Додавання та редагування тестових запитань

Дана сторінка дозволяє вводити в систему нові тестові запитання шляхом натискання кнопки «Додати».

У відповідь система відкриває форму, в яку потрібно внести наступні дані:

- База тестів, до якої належить дане питання (перелік доступних баз задається на відповідній сторінці)

- Текст запитання

- Кількість варіантів відповідей у питанні

- Номер вірної відповіді

- Варіанти відповідей від 1 до 8, кожна у відповідному полі. Якщо варіантів відповідей менше 8, то решту полів залишаємо порожніми

Після заповнення всіх необхідних даних та натискання кнопки «Зберегти» система вносить тестове запитання в базу даних. Також система передбачає редагування введених тестових запитань у випадку виявлених помилок. Для цього необхідно натиснути кнопку «Редагувати» біля відповідного запитання, внести корективи та натиснути кнопку «Зберегти».

Також передбачено видалити запитання натиснувши кнопку «Видалити» біля відповідного типу. Видалене запитання можна відновити натиснувши кнопку «Відновити».

11. Додавання та редагування інструкторів

Дана сторінка дозволяє вносити в систему нових інструкторів шляхом натискання кнопки «Додати». У відповідь система відкриває форму, в яку потрібно внести наступні дані:

- Прізвище інструктора

- Ім’я інструктора

- По батькові інструктора (за наявності)

- Резюме інструктора

- Стать інструктора

- Дата народження інструктора

- Електронна пошта інструктора

- Освіта інструктора

- Місце роботи інструктора

- Посада інструктора

Після заповнення всіх необхідних даних та натискання кнопки «Зберегти» система вносить інструктора в базу даних.

Також система передбачає редагування даних інструктора у випадку виявлених помилок або зміни інформації. Для цього необхідно натиснути кнопку «Редагувати» біля відповідного інструктора, внести корективи та натиснути кнопку «Зберегти».

Також передбачено видалити інструктора натиснувши кнопку «Видалити» біля відповідного інструктора. Видаленого інструктора можна відновити натиснувши кнопку «Відновити».

Розробка архітектури системи є вирішальним кроком у створенні програмного забезпечення, адже адекватний вибір архітектурної структури сприяє зручності масштабування системи та інтеграції нових можливостей.

3.2 Розробка структури бази даних

Для забезпечення високої продуктивності системи, важливо мати ефективний механізм для зберігання об'ємних даних у безпечному сховищі. Враховуючи необхідність обробки великого об’єму даних, які потребують значних ресурсів системи, критично важливо розробити міцну та надійну базу даних.

Система має відповідати за архівацію даних, що стосуються користувачів та їхньої активності в системі, щоб забезпечити ефективність роботи. Це вимагає розробки структурованої схеми бази даних, яка описує колекції та поля, для оптимального контролю та збереження критично важливої інформації. На рис. 3.3 представлено структуру бази даних.

У реляційній базі даних MySQL не всі поля є обов'язковими для кожної таблиці. Тип кожного поля також вказаний у дужках поряд з його назвою.

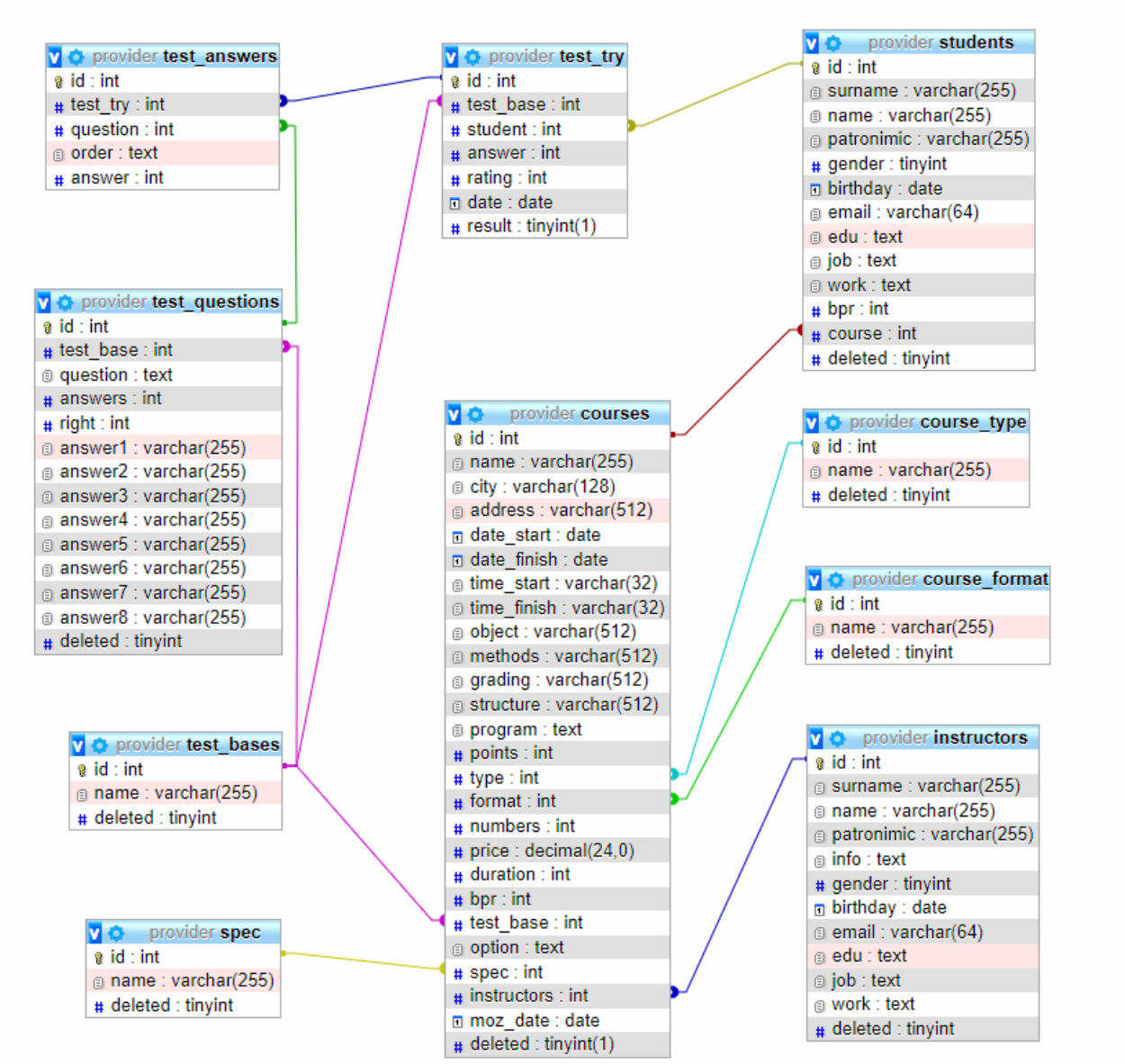


Рисунок 3.3 – Схема структури бази даних

Основна структура бази даних складається з наступних таблиць: "courses", "students", "test_bases", "test_questions", "test_try", "test_answers", "instructors", "course_type", "course_format", "spec". Кожна таблиця відіграє ключову роль у функціонуванні системи та зберіганні відповідної інформації.

В кожній таблиці існує поле "id". При створенні запису заповнення цього поля не є обов'язковим, адже система баз даних автоматично присвоює кожному елементу унікальний ідентифікатор. Проте, це поле включене до схеми кожної таблиці, щоб надати зрозуміле уявлення про структуру записів. Це допомагає краще осмислити організацію даних у базі.

Також необхідно відмітити, що на рівні архітектури в системі вирішено не передбачати видалення записів з бази даних. Замість цього в таблицях бази даних на відповідний запис передбачено встановлення позначки «Видалено». Такий механізм має низку переваг:

1. Встановлення позначки «Видалено» дозволяє зберегти історичну цілісність даних, що може бути важливим для аудиту, аналізу або відновлення даних. Повне видалення запису може призвести до втрати важливих відомостей.

2. У базі даних використовуються зв'язки між таблицями. Видалення запису може порушити ці зв'язки та призвести до неконсистентності даних. Встановлення позначки «Видалено» дозволяє зберегти всі зв'язки.

3. Адміністратор або система можуть помилково видалити записи. З позначкою «Видалено» ці записи можна легко відновити, просто знявши позначку, тоді як фізичне видалення вимагатиме складніших процедур відновлення з резервних копій.

4. В реляційній базі даних, видалення записів спричиняє перебудову індексів та фрагментацію таблиць, що знижує продуктивність. Просте встановлення позначки «Видалено» вимагає менше ресурсів і швидше виконується.

5. В деяких випадках необхідно проводити детальний аудит змін даних. Встановлення позначки «Видалено» дозволяє зберігати всі операції над даними, включаючи видалення, що спрощує аудит і контроль.

6. Навіть видалені дані можуть бути корисними для різних аналітичних цілей. Встановлення позначки «Видалено» дозволяє враховувати такі дані в аналітичних звітах.

Таким чином, використання позначки «Видалено» замість фактичного видалення запису надає більшу гнучкість і надійність при роботі з даними, тому саме цей механізм вирішено застосувати в розроблювальній системі.

В таблиці "courses", яка містить інформацію про курси, представлені наступні поля:

- "id" – ціле число, обов’язкове поле, містить унікальний ідентифікатор (автоінкремент), є первинним ключем, зв’язане зовнішнім ключем з полем "course" таблиці "students".

- "name" – строка, довжина до 255 символів, обов’язкове поле, містить назву курсу.

- "city" – строка, довжина до 128 символів, обов’язкове поле, містить назву міста в якому проводиться курс.

- "address" – строка, довжина до 512 символів, обов’язкове поле, містить адресу, за якою проводиться курсу.

- "date_start" – дата, обов’язкове поле, містить дату початку курсу.

- "date_finish" – дата, обов’язкове поле, містить дату завершення курсу (для одноденних курсів співпадає з датою початку курсу).

- "time_start" – строка, довжина до 32 символів, обов’язкове поле, містить час початку курсу.

- "time_finish" – строка, довжина до 32 символів, обов’язкове поле, містить час завершення курсу.

- "object" – строка, довжина до 512 символів, обов’язкове поле, містить мету проведення курсу.

- "methods" – строка, довжина до 512 символів, обов’язкове поле, містить форми та методи проведення курсу.

- "grading" – строка, довжина до 512 символів, обов’язкове поле, містить методи контролю знань під час курсу.

- "structure" – строка, довжина до 512 символів, обов’язкове поле, містить опис зведеної структури курсу.

- "program" – строка, довжина до 65535 символів, обов’язкове поле, містить погодинну програму проведення курсу.

- "points" – ціле число, обов’язкове поле, містить кількість балів БПР, що нараховуються учасникам курсу.

- "type" – ціле число, обов’язкове поле, містить унікальний ідентифікатор типу курсу, є зовнішній ключ на таблицю "course_type".

- "format" – ціле число, обов’язкове поле, містить унікальний ідентифікатор формату курсу, є зовнішній ключ на таблицю "course_format".

- "numbers" – ціле число, обов’язкове поле, містить максимальну кількість учасників курсу.

- "price"" – число з комою, обов’язкове поле, містить ціну курсу у гривнях.

- "duration" " – ціле число, обов’язкове поле, містить тривалість курсу у годинах.

- "bpr" " – ціле число, обов’язкове поле, містить номер курсу в системі БПР (призначається Центром тестування МОЗ після реєстрації курсу).

- "test_base" – ціле число, обов’язкове поле, містить унікальний ідентифікатор бази тестів для курсу, є зовнішній ключ на таблицю "test_bases".

- "option" – строка, довжина до 65535 символів, обов’язкове поле, містить опції тестування у форматі JSON.

- "spec" – строка, довжина до 65535 символів, обов’язкове поле, містить список унікальних ідентифікаторів спеціальностей курсу у форматі JSON.

- "instructors" – строка, довжина до 65535 символів, обов’язкове поле, містить список унікальних ідентифікаторів інструкторів курсу у форматі JSON.

- "moz_date" – дата, не обов’язкове поле, містить дату подання звіту по курсу в МОЗ.

- "deleted" – Boolean, обов’язкове поле, містить ознаку видалення запису, за замовчуванням має значення "false".

В таблиці "course_format", яка містить інформацію про формати курсів, представлені наступні поля:

- "id" – ціле число, обов’язкове поле, містить унікальний ідентифікатор (автоінкремент), є первинним ключем, зв’язане зовнішнім ключем з полем "format" таблиці "courses".

- "name" – строка, довжина до 255 символів, обов’язкове поле, містить назви форматів курсу.

- "deleted" – Boolean, обов’язкове поле, містить ознаку видалення запису, за замовчуванням має значення "false".

В таблиці "course_type", яка містить інформацію про типи курсів, представлені наступні поля:

- "id" – ціле число, обов’язкове поле, містить унікальний ідентифікатор (автоінкремент), є первинним ключем, зв’язане зовнішнім ключем з полем "type" таблиці "courses".

- "name" – строка, довжина до 255 символів, обов’язкове поле, містить назви типів курсу.

- "deleted" – Boolean, обов’язкове поле, містить ознаку видалення запису, за замовчуванням має значення "false".

В таблиці "spec", яка містить інформацію про медичні / фармацевтичні спеціальності, представлені наступні поля:

- "id" – ціле число, обов’язкове поле, містить унікальний ідентифікатор (автоінкремент), є первинним ключем.

- "name" – строка, довжина до 255 символів, обов’язкове поле, містить назви медичних / фармацевтичних спеціальностей.

- "deleted" – Boolean, обов’язкове поле, містить ознаку видалення запису, за замовчуванням має значення "false".

В таблиці "instructors", яка містить інформацію про інструкторів, представлені наступні поля:

- "id" – ціле число, обов’язкове поле, містить унікальний ідентифікатор (автоінкремент), є первинним ключем.

- "surname" – строка, довжина до 255 символів, обов’язкове поле, містить прізвище інструктора.

- "name" – строка, довжина до 255 символів, обов’язкове поле, містить і’мя інструктора.

- "patronimic" – строка, довжина до 255 символів, не обов’язкове поле, містить по батькові інструктора.

- "info" – строка, довжина до 65535 символів, обов’язкове поле, містить резюме інструктора.

- "gender" – ціле число, обов’язкове поле, містить стать інструктора (1 – чоловіча, 2 – жіноча).

- "birthday" – дата, обов’язкове поле, містить дату народження інструктора.

- "email" – строка, довжина до 255 символів, обов’язкове поле, містить електронну адресу інструктора.

- "edu" – строка, довжина до 65535 символів, обов’язкове поле, містить освіту інструктора.

- "job" – строка, довжина до 65535 символів, обов’язкове поле, містить місце роботи інструктора.

- "work" – строка, довжина до 65535 символів, обов’язкове поле, містить посаду інструктора на місці роботи.

- "deleted" – Boolean, обов’язкове поле, містить ознаку видалення запису, за замовчуванням має значення "false".

В таблиці "students", яка містить інформацію про учасників курсів, представлені наступні поля:

- "id" – ціле число, обов’язкове поле, містить унікальний ідентифікатор (автоінкремент), є первинним ключем, зв’язане зовнішнім ключем з полем "student" таблиці "test_try".

- "surname" – строка, довжина до 255 символів, обов’язкове поле, містить прізвище учасника курсу.

- "name" – строка, довжина до 255 символів, обов’язкове поле, містить і’мя учасника курсу.

- "patronimic" – строка, довжина до 255 символів, не обов’язкове поле, містить по батькові учасника курсу.

- "gender" – ціле число, обов’язкове поле, містить стать учасника курсу (1 – чоловіча, 2 – жіноча).

- "birthday" – дата, обов’язкове поле, містить дату народження учасника курсу.

- "email" – строка, довжина до 255 символів, обов’язкове поле, містить електронну адресу учасника курсу.

- "edu" – строка, довжина до 65535 символів, обов’язкове поле, містить освіту учасника курсу.

- "job" – строка, довжина до 65535 символів, обов’язкове поле, містить місце роботи учасника курсу.

- "work" – строка, довжина до 65535 символів, обов’язкове поле, містить посаду учасника курсу на місці роботи.

- "bpr" – ціле число, обов’язкове поле, містить унікальний ідентифікатор учасника на курсі (розраховується системою, по принципу автоінкременту в рамках кожного окремого курсу).

- "course" – ціле число, обов’язкове поле, містить унікальний ідентифікатор курсу на якому навчався учасник, є зовнішній ключ на таблицю "courses".

- "deleted" – Boolean, обов’язкове поле, містить ознаку видалення запису, за замовчуванням має значення "false".

В таблиці "test_bases", яка містить інформацію про бази тестів, представлені наступні поля:

- "id" – ціле число, обов’язкове поле, містить унікальний ідентифікатор (автоінкремент), є первинним ключем, зв’язане зовнішнім ключем з полями "test_base" таблиці "courses", "test_base" таблиці "test_questions" та "test_base" таблиці "test_try".

- "name" – строка, довжина до 255 символів, обов’язкове поле, містить назви баз тестів.

- "deleted" – Boolean, обов’язкове поле, містить ознаку видалення запису, за замовчуванням має значення "false".

В таблиці "test_questions", яка містить інформацію про питання тестів, представлені наступні поля:

- "id" – ціле число, обов’язкове поле, містить унікальний ідентифікатор (автоінкремент), є первинним ключем, зв’язане зовнішнім ключем з полем "question" таблиці "test_answers".

- "test_base" – ціле число, обов’язкове поле, містить унікальний ідентифікатор бази тестів до якої належить питання, є зовнішній ключ на таблицю "test_bases".

- "question" – строка, довжина до 65535 символів, обов’язкове поле, містить текст питання.

- "answers" – ціле число, обов’язкове поле, містить кількість варіантів відповідей в питанні (від 2 до 8).

- "right" – ціле число, обов’язкове поле, містить номер варіанта правильної відповіді в питанні.

- "answer1", "answer2", "answer3", "answer4", "answer5", "answer6", "answer7", "answer8" – строки, довжина до 255 символів, обов’язкові поля, містять тексти варіантів відповідей в питанні, відповіді понад значення поля "answers" за замовчанням порожні.

- "deleted" – Boolean, обов’язкове поле, містить ознаку видалення запису, за замовчуванням має значення "false".

В таблиці "test_try", яка містить інформацію про спроби проходження тестування учасниками курсу, представлені наступні поля:

- "id" – ціле число, обов’язкове поле, містить унікальний ідентифікатор (автоінкремент), є первинним ключем, зв’язане зовнішнім ключем з полем "test_try" таблиці "test_answers".

- "test_base" – ціле число, обов’язкове поле, містить унікальний ідентифікатор бази тестів до якої належить питання, є зовнішній ключ на таблицю "test_bases".

- "student" – ціле число, обов’язкове поле, містить унікальний ідентифікатор учасника курсу, є зовнішній ключ на таблицю "students".

- "answer" – ціле число, обов’язкове поле, містить кількість правильних відповідей, які дав учасник під час тестування.

- "rating" – ціле число, обов’язкове поле, містить кількість правильних відповідей, які мінімально необхідні для проходження тестування.

- "date" – дата та час, обов’язкове поле, містить дату та час спроби проходження тестування.

- "result" – Boolean, обов’язкове поле, містить ознаку успішності спроби тестування, за замовчуванням має значення "false".

Дана таблиця є службовою, дані в неї вносяться системою автоматично – відповідно поле "deleted" не передбачено.

В таблиці "test_answers", яка містить інформацію детальну про кожну спробу проходження тестування учасниками курсу, представлені наступні поля:

- "id" – ціле число, обов’язкове поле, містить унікальний ідентифікатор (автоінкремент), є первинним ключем.

- "test_try" – ціле число, обов’язкове поле, містить унікальний ідентифікатор спробу проходження тестування, є зовнішній ключ на таблицю "test_try".

- "question" – ціле число, обов’язкове поле, містить унікальний ідентифікатор питання, є зовнішній ключ на таблицю "test_questions".

- "order" – строка, довжина до 65535 символів, обов’язкове поле, містить порядок унікальних ідентифікаторів варіантів відповідей, в якому вони були задані учаснику під час тестування (система завжди перемішує варіанти відповідей у випадковому порядку) у форматі JSON.

- "answer" – ціле число, обов’язкове поле, містить номер варіанта правильної відповіді, який дав учасник під час тестування.

Дана таблиця також є службовою, дані в неї вносяться системою автоматично – відповідно поле "deleted" не передбачено.

База даних MySQL була обрана для зберігання даних, оскільки вона ідеально підходить для потреб проекту та підтримує всі потрібні типи даних.

Перевагою MySQL є її спроможність ефективно працювати з об'ємними та різнорідними наборами даних, а також її адаптивність до структури даних.

3.3 Розробка архітектури системи

Використання PHP є ключовим у створенні системи, що дозволяє реалізувати необхідні компоненти з урахуванням архітектурних вимог. Це забезпечує гладку взаємодію між системою та клієнтом. Система відповідає за обробку даних, отриманих від клієнта, та виконання запитів до бази даних. Структура каталогів представлена на рис. 3.4.

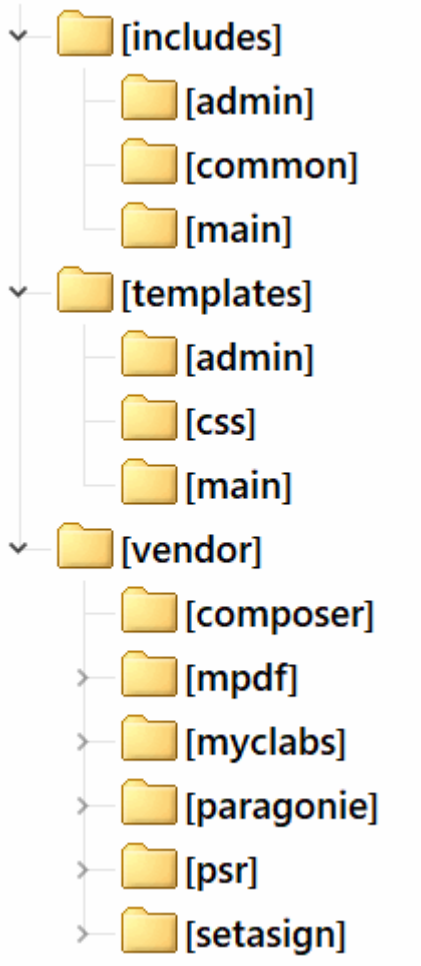


Рисунок 3.4 – Структура каталогів системи

Архітектура системи розподілена на окремі папки – "includes", "templates" та "vendor", які розміщено в кореневій папці. Також в кореневій папці розміщені основні індексні файли – "index.php" та "admin.php", файли з

налаштуваннями composer – "composer.json" та "composer.lock", а також файл з конфігурацією web-сервера apache – ".htaccess" (рис. 3.5).

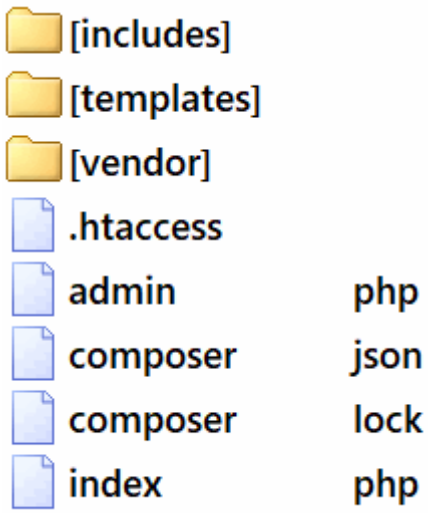


Рисунок 3.5 – Список файлів та папок в кореневій папці системи

Індексний файл "index.php" служить центральним вузлом, який приймає запити від клієнта при перегляді клієнтської частини системи та розподіляє їх для подальшої обробки.

Контролери клієнтської частини розміщені в папці "includes/main" і містять обробники запитів для кожної сторінки клієнтської частини системи (рис. 3.6).

Вони виконують такі основні операції:

- main.php – Перегляд головної сторінки
- courses_list.php – Перегляд списку курсів.
- course_detail.php – Перегляд детальної інформації про курси.
- docs.php – Перегляд документів.
- contacts.php – Перегляд контактів.
- verify.php – Перевірка сертифікатів.
- course_form.php – Подання заявки на реєстрацію на курс.

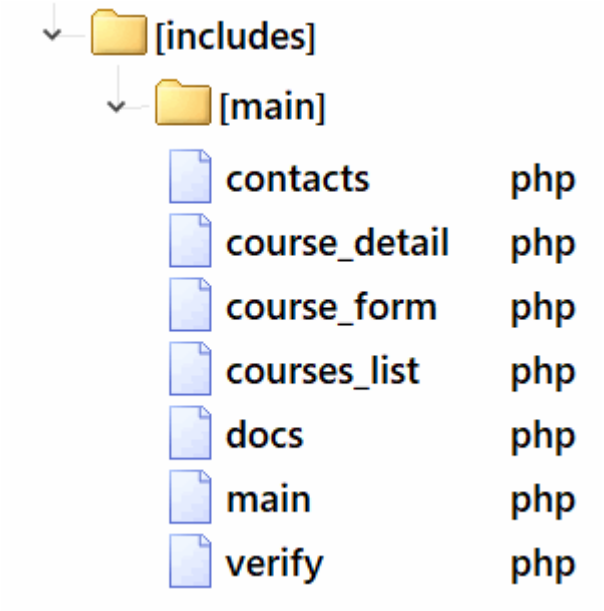


Рисунок 3.6 – Список файлів в папці "includes/main"

Шаблони сторінок клієнтської частини розміщені, відповідно в папці "templates/main" і містять html код для кожної сторінки клієнтської частини системи (рис. 3.7).

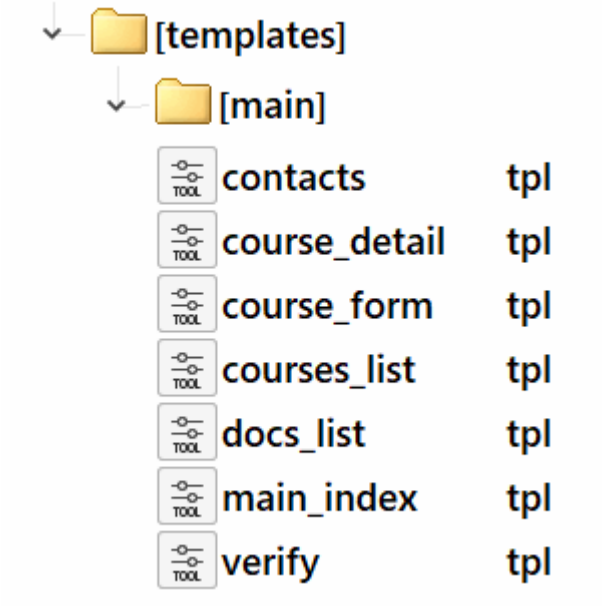


Рисунок 3.7 – Список файлів в папці "templates/main"

Індексний файл "admin.php" служить центральним вузлом, який приймає запити від клієнта при перегляді адміністративної частини системи та розподіляє їх для подальшої обробки.

Контролери адміністративної частини розміщені в папці "includes/admin" і містять обробники запитів для кожної сторінки адміністративної частини системи (рис. 3.8).

Вони виконують такі основні операції:

- courses_edit.php – Редагування інформації про курс.

- courses_list.php – Перегляд списку курсів.

- courses_moz_register.php – Перегляд інформації для реєстрації курсу в МОЗ.

- courses_moz_report.php – Перегляд інформації для подачі звіту по курсу в МОЗ.

- courses _tests.php – Налаштування опцій тестування для курсу.

- course_format_edit.php – Редагування формату курсу.

- course_format_list.php – Перегляд списку форматів курсів.

- course_type_edit.php – Редагування типу курсу.

- course_type_list.php – Перегляд списку типів курсів.

- instructors_edit.php – Редагування інформації про інструктора.

- instructors_list.php – Перегляд списку інструкторів.

- spec_edit.php – Редагування інформації про медичну / фармацевтичну спеціальність.

- spec_list.php – Перегляд списку медичних / фармацевтичних спеціальністей.

- students_cert.php – Перегляд сертифікату учасника курсу.

- students_edit.php – Редагування інформації учасника курсу.

- students_list.php – Перегляд списку учасників курсу.

- test_bases_edit.php – Редагування інформації про базу тестів.

- test_bases_list.php – Перегляд списку баз тестів.

- test_questions_edit.php – Редагування тестового питання.

- test_questions_list.php – Перегляд списку тестових питань.

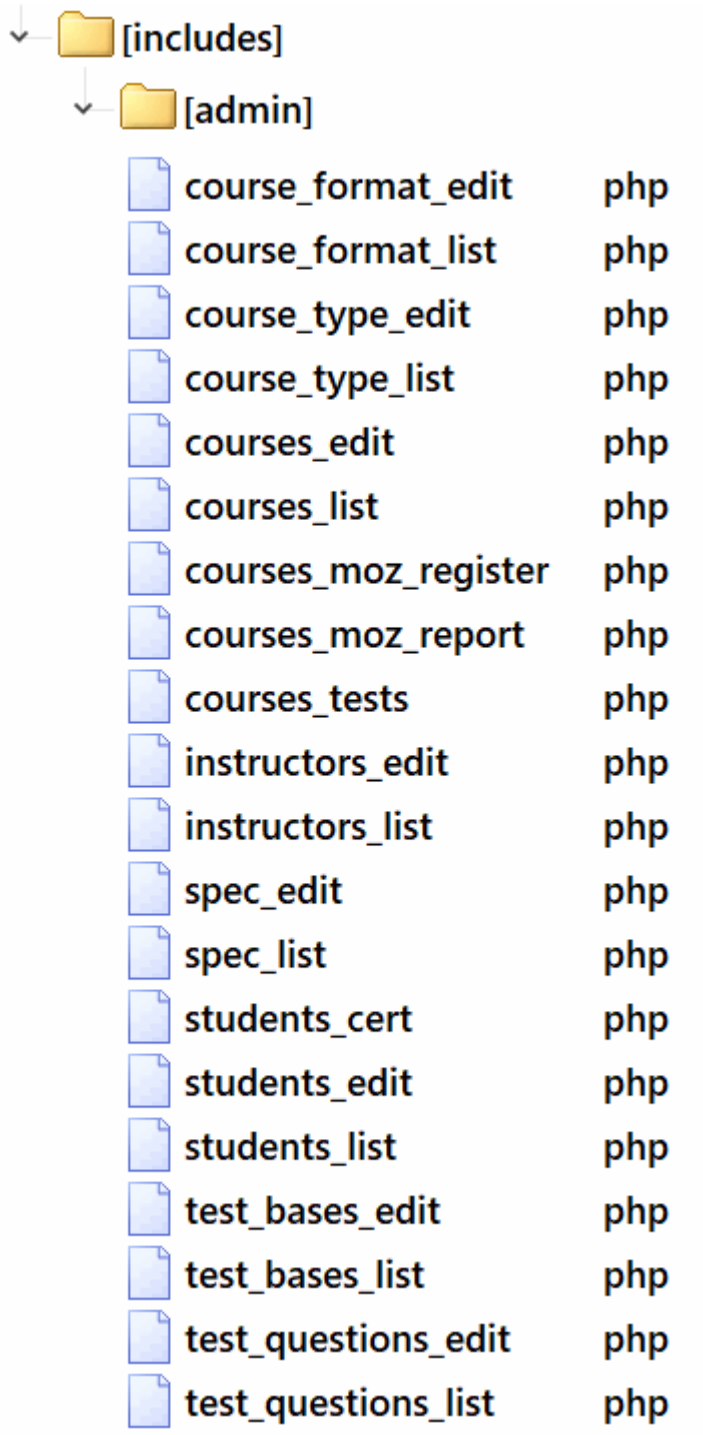


Рисунок 3.8 – Список файлів в папці "includes/admin"

Шаблони сторінок клієнтської частини розміщені, відповідно в папці "templates/admin" і містять html код для кожної сторінки клієнтської частини системи (рис. 3.9).

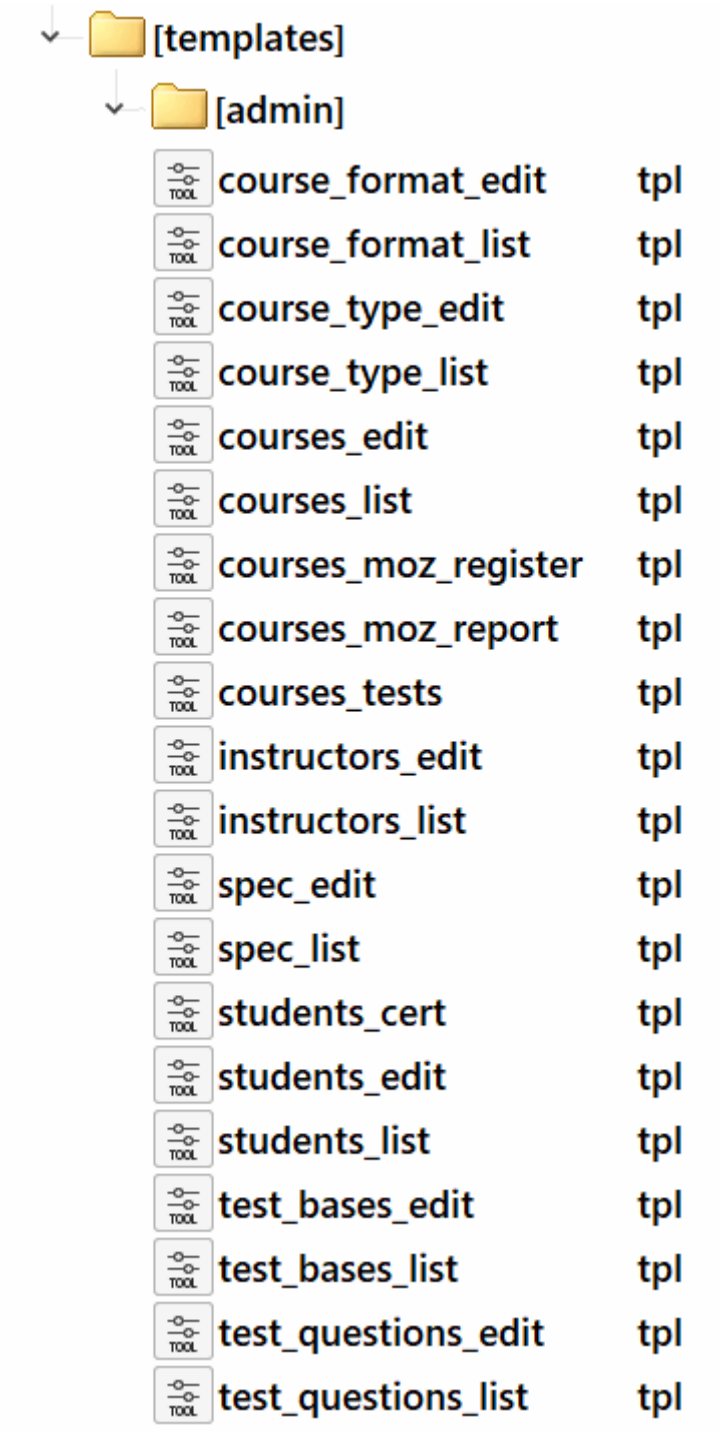


Рисунок 3.9 – Список файлів в папці "templates/admin"

Також в папці "includes/common" розміщені файли, які використовуються всіма контролерами (рис. 3.10).

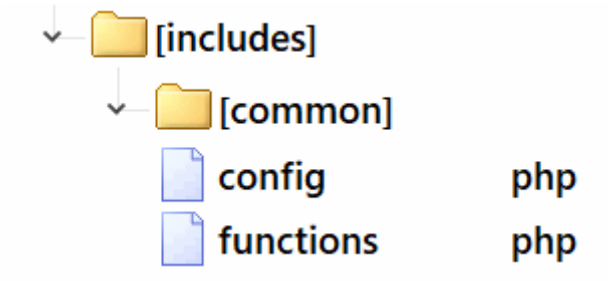


Рисунок 3.10 – Список файлів в папці "includes/common"

Вони виконують такі основні операції:

- config.php – Містить конфігураційні налаштування системи.

- functions.php – Містить загальні методи та функції системи.

Відповідно в папці "templates/css" розміщені файли стилів. Файл "style.css" містить стилі, які використовуються в усіх шаблонах відображення, а файл "cert.css" містить стилі до шаблону сертифікату (рис. 3.11).

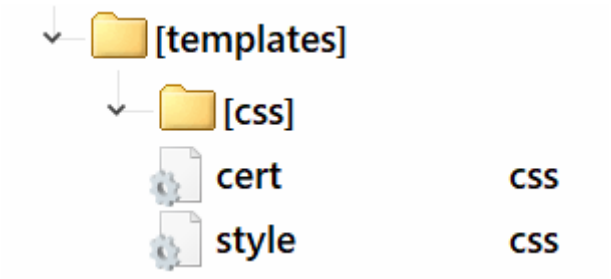


Рисунок 3.11 – Список файлів в папці "templates/css"

Також в кореневій папці розміщена папка "vendor", в які розташовані файли composer.

3.3 Розробка інтерфейсу користувача

На головній сторінці можна побачити строку меню, через яку доступні основні сторінки користувацької частини системи (рис. 3.12).

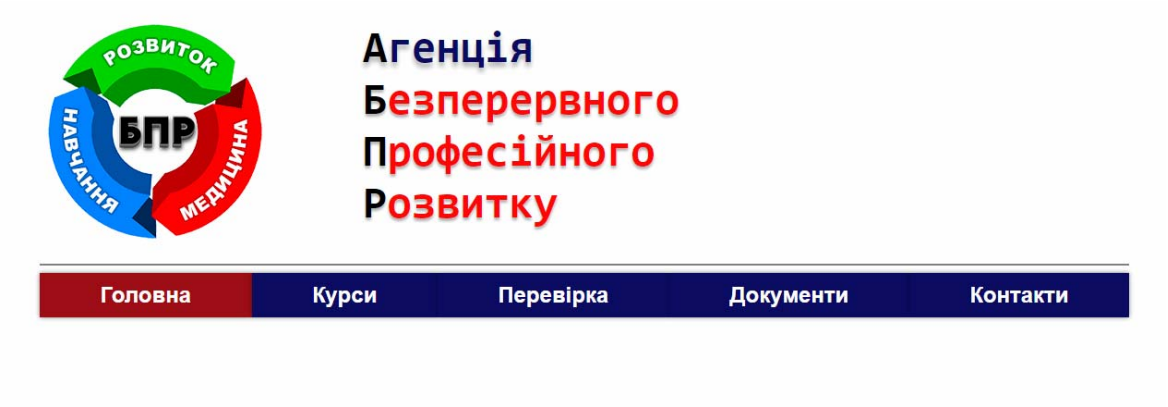


Рисунок 3.12 – Інтерфейс головної сторінки

На сторінці "Курси" доступний перегляд список актуальних курсів (рис. 3.13).

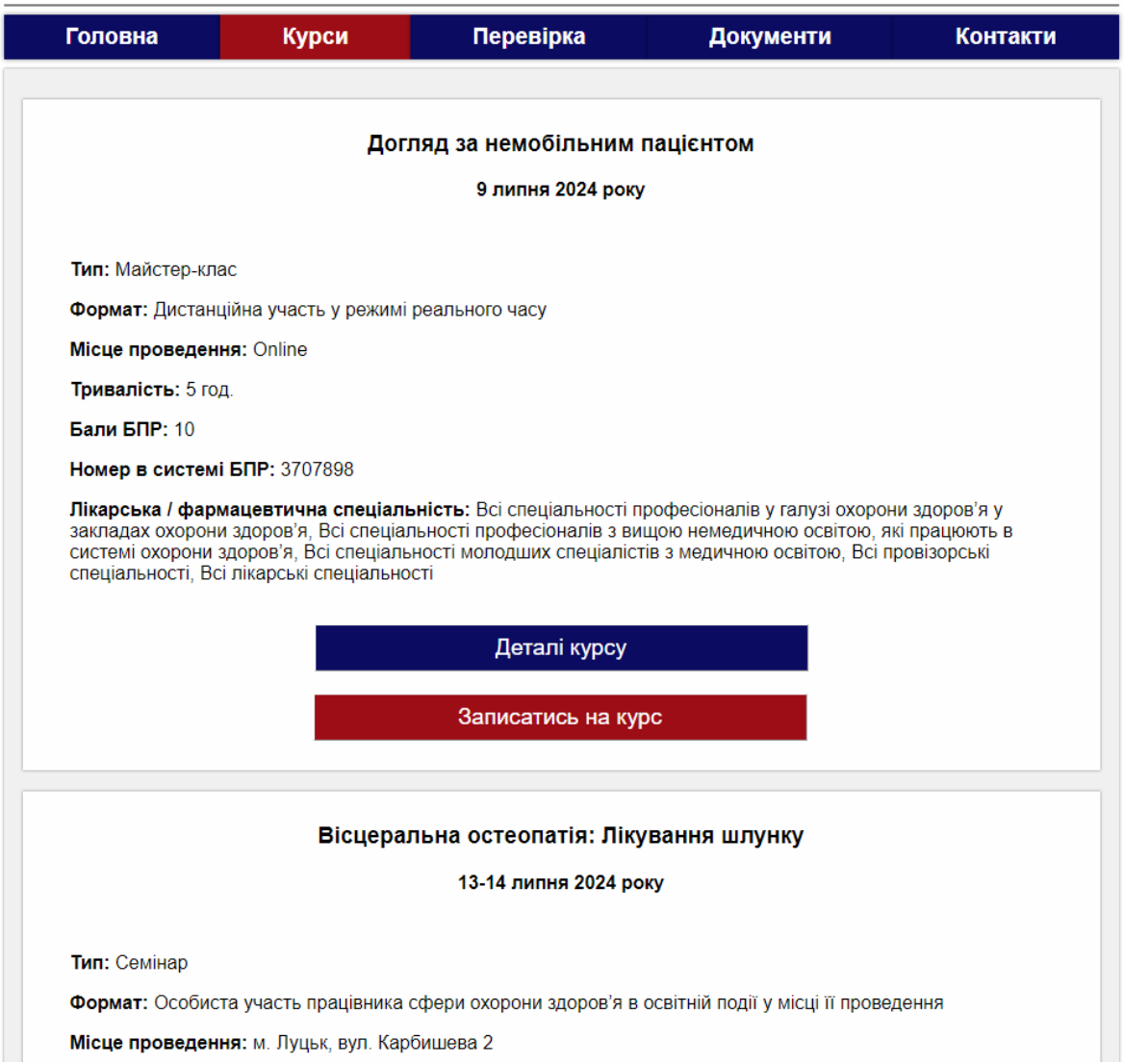


Рисунок 3.13 – Інтерфейс сторінки "Курси"

Підтримується розділення списку на сторінки. На одну сторінку виводиться до 10 курсів. Механізм розділення на сторінки є універсальним та застосовується на всіх сторінках системи (рис. 3.14).

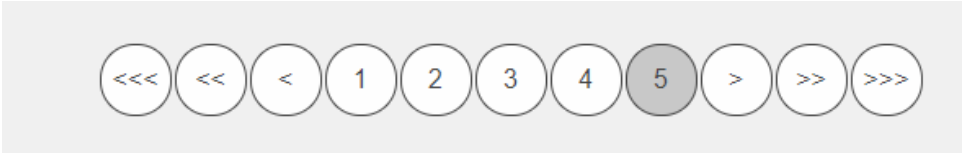
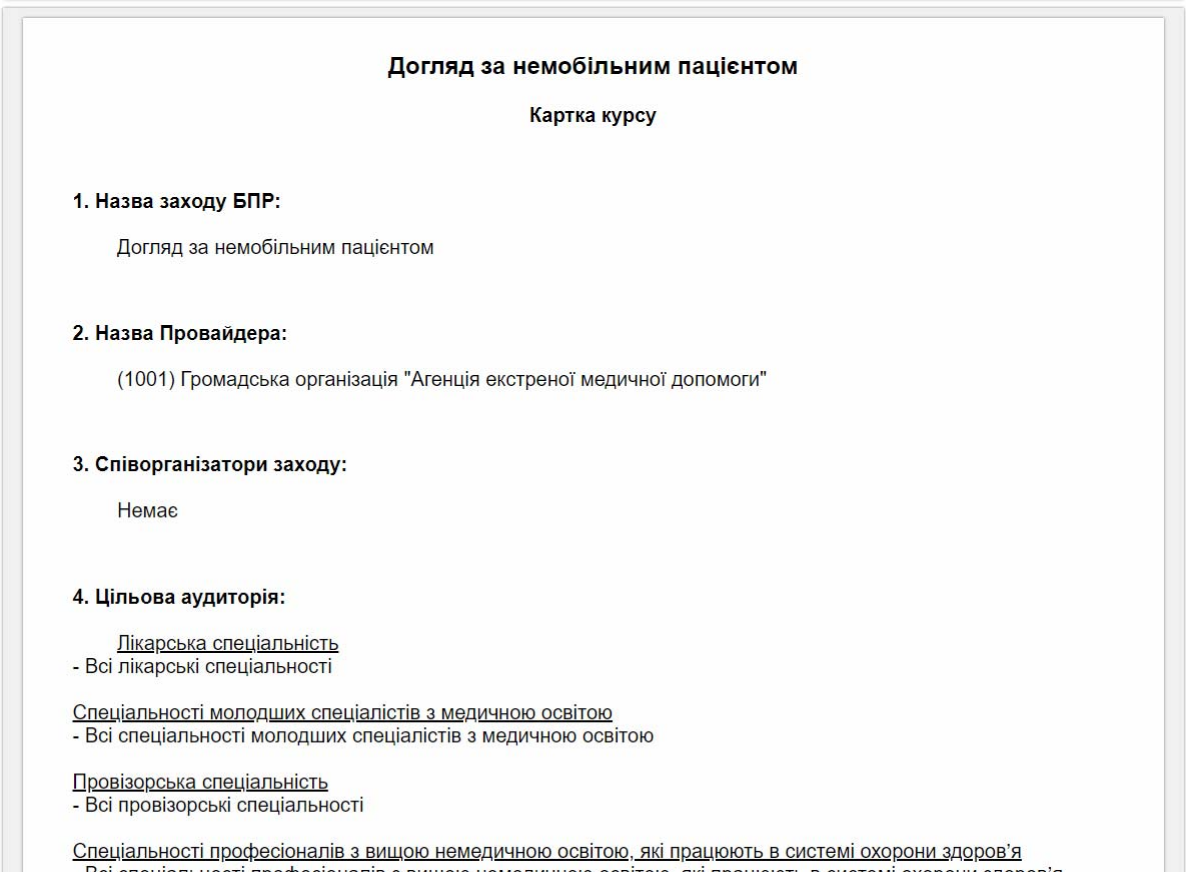


Рисунок 3.14 – Інтерфейс механізму розділення на сторінки

Можна переглянути основну інформацію про курс після якої розташовані кнопки "Деталі курсу" та "Записатись на курс".

Натискання на кнопку "Деталі курсу" перенаправляє на сторінку з детальною інформацією про курс. В постанові 725 інформацію про курс називають "Картка курсу" [1] і затверджують перелік даних, які мають бути відображені на сторінці (рис. 3.15).



Догляд за немобільним пацієнтом

Картка курсу

1. Назва заходу БПР:

Догляд за немобільним пацієнтом

2. Назва Провайдера:

(1001) Громадська організація "Агенція екстреної медичної допомоги"

3. Співорганізатори заходу:

Немає

4. Цільова аудиторія:

Лікарська спеціальність
- Всі лікарські спеціальності

Спеціальності молодших спеціалістів з медичною освітою
- Всі спеціальності молодших спеціалістів з медичною освітою

Провізорська спеціальність
- Всі провізорські спеціальності

Спеціальності професіоналів з вищою немедичною освітою, які працюють в системі охорони здоров'я

Рисунок 3.15 – Інтерфейс сторінки "Деталі курсу"

Натискання на кнопку "Записатись на курс" відкриває сторінку з реєстраційною формою (рис. 3.16).

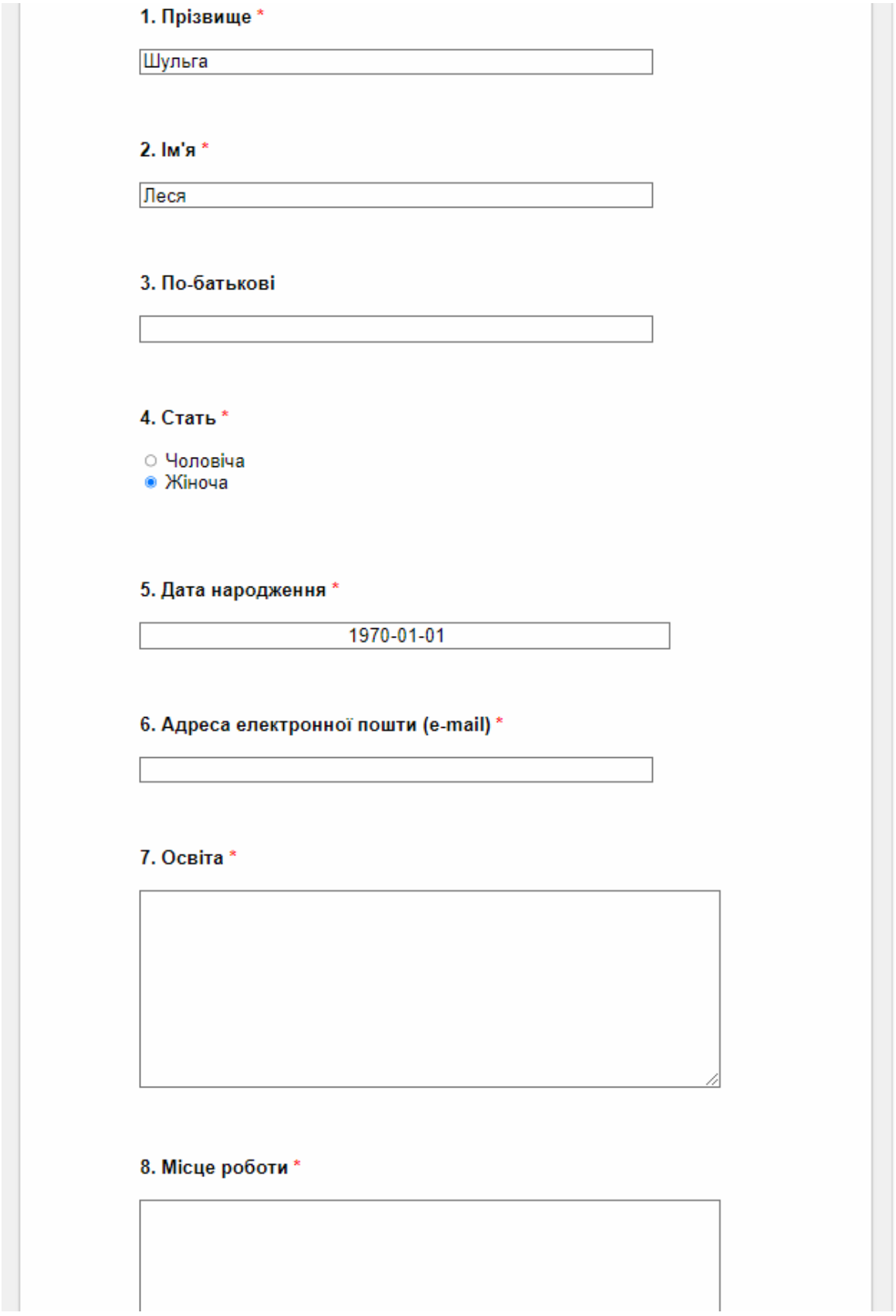


Рисунок 3.16 – Інтерфейс сторінки "Записатись на курс"

Необхідно заповнити всі обов’язкові поля (відмічені зірочкою) та натиснути кнопку "Відправити". Система вносить інформацію учасника в базу даних та виводить відповідне підтвердження.

На сторінці "Перевірка сертифікату" відображається поле для вводу номеру сертифікату (рис. 3.17).

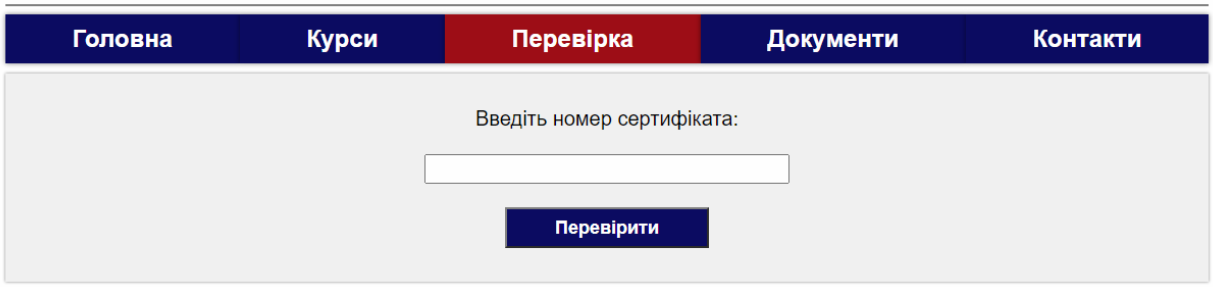


Рисунок 3.17 – Інтерфейс сторінки "Перевірка сертифікату" (введення номера сертифікату)

Після введення номеру та натискання на кнопку "Перевірити" система відображає детальну інформацію про відповідний сертифікат (рис. 3.18).

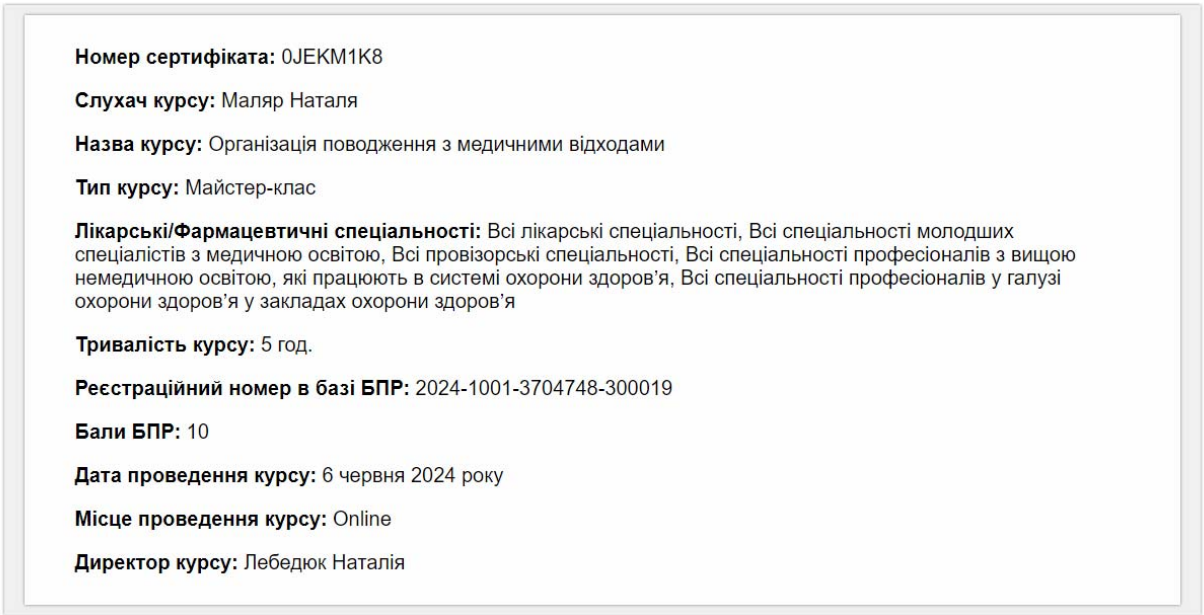


Рисунок 3.18 – Інтерфейс сторінки "Перевірка сертифікату" (перегляд детальної інформації про сертифікат)

На сторінці "Документи" відображається список завантажених документів (рис. 3.19).

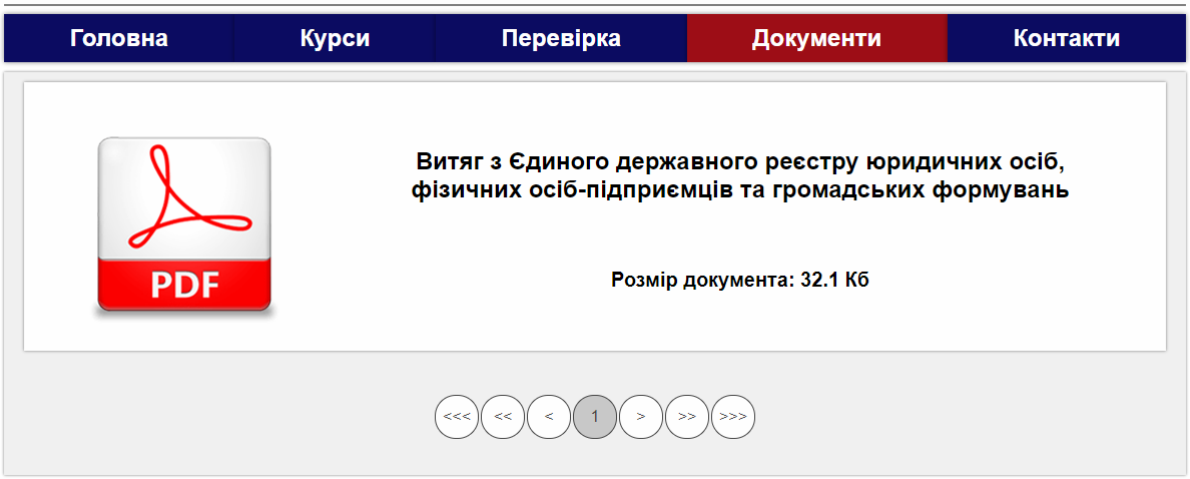


Рисунок 3.19 – Інтерфейс сторінки "Документи" (перегляд списку документів)

Натискання на назву документа відкриває сторінку перегляду документу (рис. 3.20).

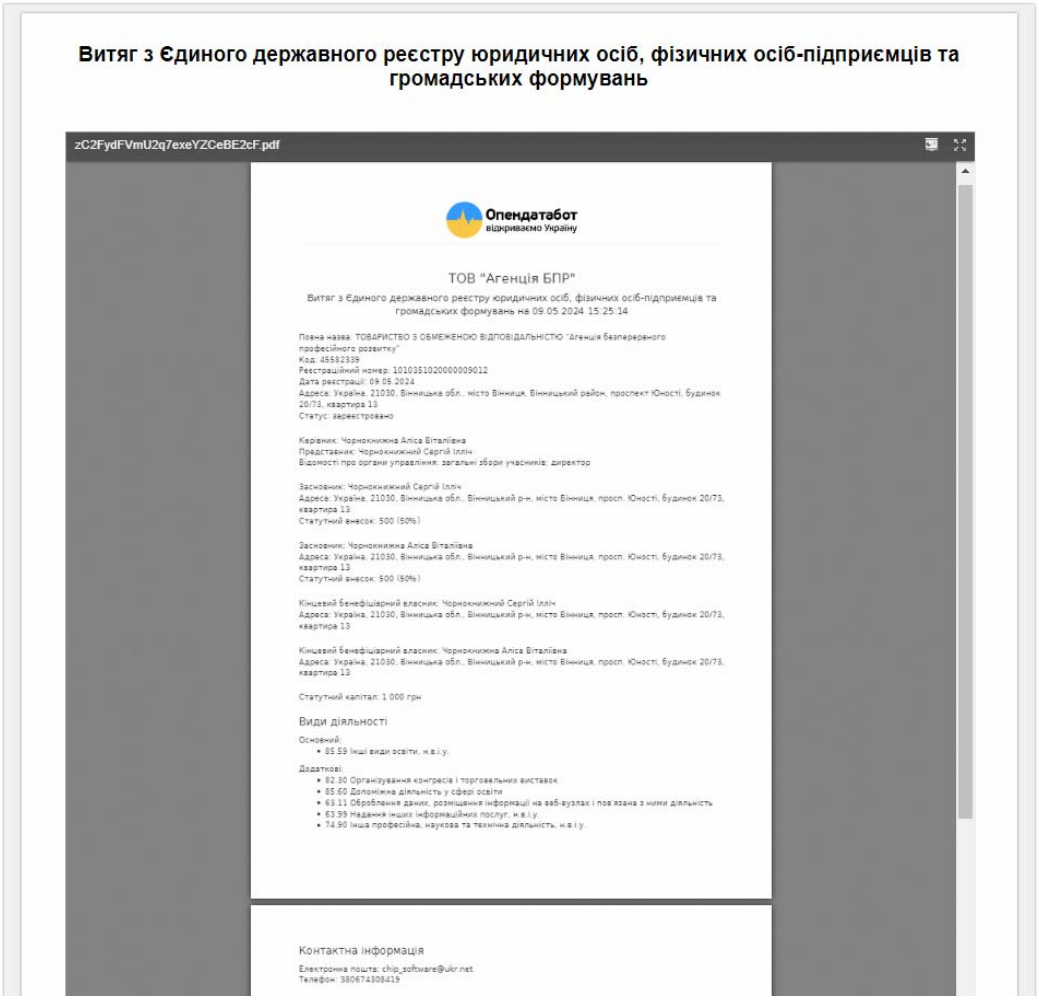


Рисунок 3.20 – Інтерфейс сторінки "Документи" (перегляд документу)

На сторінці "Контакти" відображається контактна інформація провайдера (рис. 3.21).

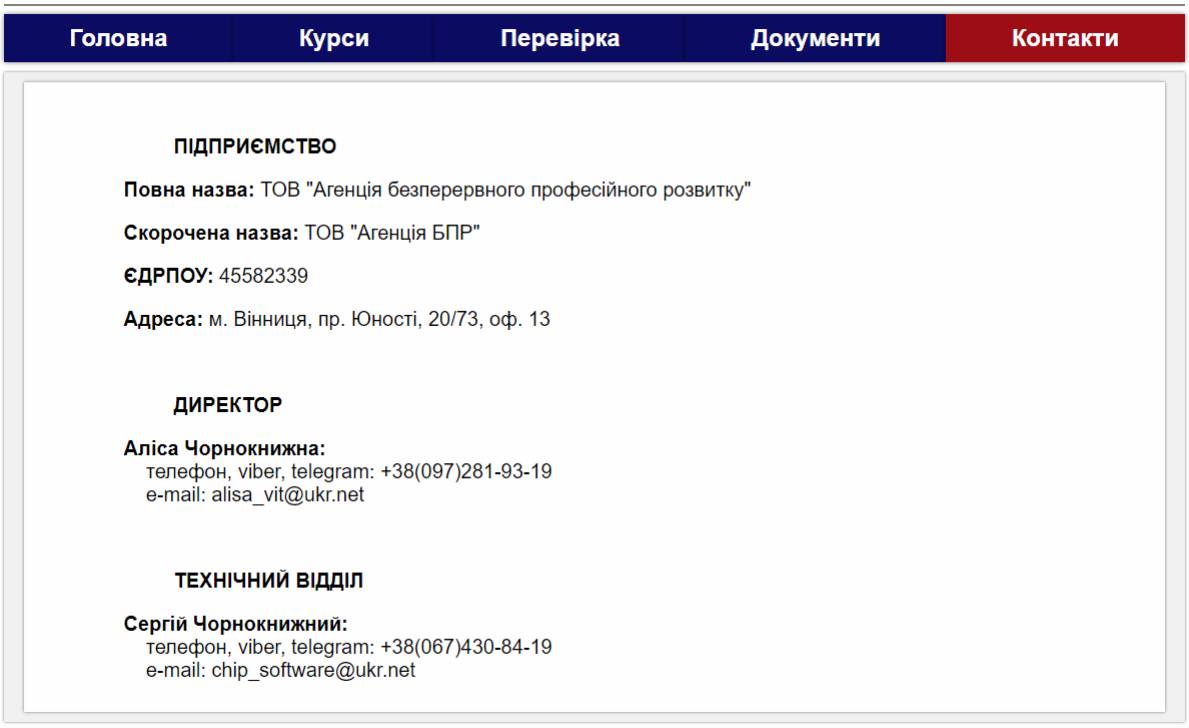


Рисунок 3.21 – Інтерфейс сторінки "Контакти"

Аналогічним чином побудований інтерфейс адміністративної частини системи.

3.5 Висновки до розділу 3

У сфері створення програмного забезпечення особлива увага приділяється розробці та утриманню ефективних, високоякісних та стабільних програмних продуктів. Цей процес інтегрує техніки, методи та підходи з різноманітних дисциплін, включаючи комп'ютерні науки, управління проектами, математику, інженерію тощо. Розробка програмного забезпечення, як і інші технічні спеціальності, звертає увагу на аспекти якості, економічності та довговічності. Відомо, що деякі програмні продукти містять мільйони ліній коду, які мають бездоганно працювати в різних умовах.

Розробка архітектури системи є вирішальним кроком у створенні програмного забезпечення, адже адекватний вибір архітектурної структури сприяє зручності масштабування системи та інтеграції нових можливостей.

Для забезпечення високої продуктивності системи, важливо мати ефективний механізм для зберігання об'ємних даних у безпечному сховищі. Враховуючи необхідність обробки великого об’єму даних, які потребують значних ресурсів системи, критично важливо розробити міцну та надійну базу даних.

Система має відповідати за архівацію даних, що стосуються користувачів та їхньої активності в системі, щоб забезпечити ефективність роботи. Це вимагає розробки структурованої схеми бази даних, яка описує колекції та поля, для оптимального контролю та збереження критично важливої інформації.

База даних MySQL була обрана для зберігання даних, оскільки вона ідеально підходить для потреб проекту та підтримує всі потрібні типи даних. Перевагою MySQL є її спроможність ефективно працювати з об'ємними та різнорідними наборами даних, а також її адаптивність до структури даних.

Основна структура бази даних складається з наступних таблиць: "courses", "students", "test_bases", "test_questions", "test_try", "test_answers", "instructors", "course_type", "course_format", "spec". Кожна таблиця відіграє ключову роль у функціонуванні системи та зберіганні відповідної інформації.

В кожній таблиці існує поле "id". При створенні запису заповнення цього поля не є обов'язковим, адже система баз даних автоматично присвоює кожному елементу унікальний ідентифікатор. Проте, це поле включене до схеми кожної таблиці, щоб надати зрозуміле уявлення про структуру записів. Це допомагає краще осмислити організацію даних у базі.

Також необхідно відмітити, що на рівні архітектури в системі вирішено не передбачати видалення записів з бази даних. Замість цього в таблицях бази даних на відповідний запис передбачено встановлення позначки

«Видалено», що надає більшу гнучкість і надійність при роботі з даними, тому саме цей механізм вирішено застосувати в розроблювальній системі.

Використання PHP є ключовим у створенні системи, що дозволяє реалізувати необхідні компоненти з урахуванням архітектурних вимог. Це забезпечує гладку взаємодію між системою та клієнтом. Система відповідає за обробку даних, отриманих від клієнта, та виконання запитів до бази даних.

Архітектура системи розподілена на окремі папки – "includes", "templates" та "vendor", які розміщено в кореневій папці. Також в кореневій папці розміщені основні індексні файли – "index.php" та "admin.php", файли з налаштуваннями composer – "composer.json" та "composer.lock", а також файл з конфігурацією web-сервера apache – ".htaccess".

4. ЕКОНОМІЧНИЙ РОЗДІЛ

4.1 Технологічний аудит розробленої системи автоматизації процесів, що їх виконує провайдер під час організації та проведення заходів безперервного професійного розвитку медичних працівників

Як було зазначено раніше, на сучасному світовому ринку не представлено рішень для автоматизації процесів, що їх виконує провайдер під час організації та проведення заходів безперервного професійного розвитку (БПР) медичних працівників, які б враховували специфіку України. Якщо деякі варіанти таких рішень і були розроблені за замовленнями великих провайдерів для застосування у власній мережі, то для малих або середніх провайдерів подібних аналогів на українському ринку немає, а дозволити собі розробку або покупку рішень, які розраховані на великих провайдерів, вони не можуть, оскільки це досить дорого і використовувати такі рішення невигідно з боку бюджету таких провайдерів.

Тому метою виконаної нами магістерської кваліфікаційної роботи було розроблення бюджетних цільових рішень, які б задовольняли потреби невеликих фірм в автоматизації процесів провайдера безперервного професійного розвитку медичних працівників.

Зокрема, були сформовані такі етапи дій провайдера: а) формування та розміщення на власному сайті програм та картки заходу; б) здійснення реєстрації заходу в Центрі тестування МОЗ; в) проведення реєстрації учасників заходу; г) проведення тестування в електронному вигляді; д) генерація сертифікатів за встановленим МОЗ зразком; е) подання звіту про проведення заходу в Центр тестування МОЗ.

В результаті нами було розроблено систему автоматизації процесів провайдера безперервного професійного розвитку медичних працівників (БПР).

Для встановлення комерційного потенціалу розробленої нами системи автоматизації процесів провайдера було запрошено 3-х відомих експертів – доктора технічних наук, професора Бісікала О.В., доктора технічних наук, професора Дубового В.М. та кандидатку технічних наук, доценткиню Коваленко О.О.

Встановлення комерційного потенціалу розробленої нами системи автоматизації процесів провайдера було здійснено за критеріями, наведеними в табл. 4.1.

Таблиця 4.1 – Рекомендовані критерії оцінювання комерційного потенціалу будь-якої розробки і їх бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
Ринкові перспективи					
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою

Продовження таблиці 4.1

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Запрошені експерти оцінили розроблену нами систему автоматизації процесів провайдера таким чином (табл. 4.2):

Таблиця 4.2 – Результати технологічного аудиту розробленої системи автоматизації процесів провайдера (за шкалою оцінювання 0-1-2-3-4)

Критерії	Прізвище, ініціали експертів		
	Бісікало О.В.	Дубовой В.М.	Коваленко О.О.
	Бали, що їх виставили експерти:		
1	4	3	4
2	3	4	4
3	3	4	4
4	4	4	4
5	3	3	4
6	4	4	4
7	4	4	3
8	4	4	4
9	3	4	3
10	4	4	4
11	4	4	4
12	4	3	4
Сума балів	$СБ_1 = 44$	$СБ_2 = 45$	$СБ_3 = 46$
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_{1}^{3} СБ_i}{3} = \frac{44+45+46}{3} = \frac{135}{3} = 45,00$		

Встановлення комерційного потенціалу розробленої нами системи автоматизації процесів провайдера будемо здійснювати на основі рекомендацій, наведених в таблиці 4.3 [54].

Таблиця 4.3 – Рівні комерційного потенціалу будь-якої наукової розробки

Середньоарифметична сума балів $\overline{СБ}$, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0 – 10	Низький
11 – 20	Нижче середнього
21 – 30	Середній
31 – 40	Вище середнього
41 – 48	Високий

Оскільки середньоарифметична сума балів, що їх виставили експерти, складає 45 балів, то це свідчить, що розроблена нами система автоматизації процесів провайдера безперервного професійного розвитку медичних працівників (БПР) має рівень комерційного потенціалу, який вважається «високим».

Це пояснюється тим, що розроблена нами система автоматизації процесів провайдера безперервного професійного розвитку медичних працівників (БПР) базується на найсучасніших методах програмування і має невисоку вартість порівняно з іншими, спеціально розробленими великими фірмами аналогічними системами.

4.2 Розрахунок витрат на розроблення системи автоматизації процесів провайдера безперервного професійного розвитку медичних працівників

При розробленні системи автоматизації процесів провайдера безперервного професійного розвитку медичних працівників (БПР) були зроблені певні витрати.

Зокрема:

А). Основна заробітна плата $З_о$ розробників, яка визначається за формулою:

$$З_о = \frac{М}{Т_р} \cdot t \text{ грн}, \qquad (4.1)$$

де М – місячний посадовий оклад розробника, грн; приймемо, що

М = (8000…30000) грн/місяць;

$Т_р$ – число робочих днів в місяці; приймемо $Т_р$ = 24 дні;

t – число днів роботи розробників.

Зроблені розрахунки зведемо до таблиці 4.4:

Таблиця 4.4 – Основна заробітна плата розробників

Найменування посади виконавця	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на оплату праці, грн
1. Науковий керівник магістерської роботи	30000	1250	20 годин	≈ 4167
2. Магістрант-студент-виконавець	Беремо мінімальну оплату 8000 грн	333,33	86	≈ 28664
3. Консультант з економічної частини	20000	833,33	1,5 години	≈ 209 (при 6-годинному робочому дні)
4. Інші консультанти (з медичних питань, тощо)	20000	833,33	10 днів	≈ 8334
Загалом				$З_о$ = 41374 грн

Б). Додаткова заробітна плата $З_д$ розробників розраховується як (10…12)% від величини їх основної заробітної плати, тобто:

$$З_д = \alpha \cdot З_о = (0{,}1...0{,}12) \cdot З_о . \quad (4.2)$$

Приймемо, що α = 0,1. Тоді для нашого випадку отримаємо:

$$З_д = 0{,}1 \times 41374 = 4137{,}4 \approx 4138 \text{ грн.}$$

В). Нарахування на заробітну плату $НЗП_{зп}$ розробників (дослідників) розраховуються за формулою:

$$НЗП_{зп} = (З_о + З_д) \cdot \frac{\beta}{100}, \quad (4.3)$$

де β – ставка обов'язкового єдиного внеску на державне соціальне страхування, %. β = 22%. Тоді:

$$НЗН_{зп} = (41374 + 4138) \times 0{,}22 = 10012{,}64 \approx 10013 \text{ грн.}$$

Г). Амортизація основних засобів А, які використовувались під час виконання цієї роботи:

$$А = \frac{Ц \cdot Н_а}{100} \cdot \frac{Т}{12} \text{ грн}, \quad (4.4)$$

де Ц – загальна балансова вартість основних засобів, грн;

$Н_а$ – річна норма амортизаційних відрахувань. Для нашого випадку можна прийняти, що $Н_а$ = (2,5...25)%;

Т – термін використання основних засобів, місяці.

Зроблені розрахунки зведено в таблицю 4.5.

Таблиця 4.5 – Розрахунок амортизаційних відрахувань

Найменування обладнання, приміщень тощо	Балансова вартість, грн.	Норма амортизації, %	Термін використання, міс.	Величина амортизаційних відрахувань, грн
1. Комп'ютерна техніка, обладнання, принтери тощо	70000	25,0	3,5 (при 80% використанні	4083,33 ≈ 4084
2. Приміщення університету, кафедри	40000	2,5	3,5 при 40% використанні	116,67 ≈ 117
Всього				*А = 4201 грн*

Д). Витрати на матеріали М розраховуються за формулою:

$$М = \sum_{1}^{n} Н_i \cdot Ц_i \cdot К_i - \sum_{1}^{n} В_i \cdot Ц_в \quad грн., \tag{4.5}$$

де $Н_i$ – витрати матеріалу *і*-го найменування, кг; $Ц_i$ – вартість матеріалу *і*-го най-менування; $К_i$ – коефіцієнт транспортних витрат, $К_i$ = (1,1…1,15); $В_i$ – маса відходів матеріалу *і*-го найменування; $Ц_в$ – ціна відходів матеріалу *і*-го найменування; n – кількість видів матеріалів.

Е). Витрати на комплектуючі К розраховуються за формулою:

$$К = \sum_{1}^{n} Н_i \cdot Ц_i \cdot К_i \quad грн, \tag{4.6}$$

де $Н_i$ – кількість комплектуючих *і*-го виду, шт.; $Ц_i$ – ціна комплектуючих *і*-го виду; $К_i$ – коефіцієнт транспортних витрат, $К_i$ = (1,1…1,15); n – кількість видів комплектуючих.

Під час виконання магістерської кваліфікаційної роботи загальні витрати на матеріали та комплектуючі становили приблизно 2000 грн.

Ж). Витрати на силову електроенергію $В_е$ розраховуються за формулою:

$$В_е = \frac{В \cdot П \cdot Ф \cdot К_п}{К_д}, \tag{4.7}$$

де В – вартість 1 кВт-год. електроенергії, в 2024 р. В ≈ 4,5 грн/кВт;

П – установлена потужність обладнання, кВт; П = 1,25 кВт;

Ф – фактична кількість годин роботи обладнання, годин.

Приймемо, що Ф = 385 годин;

$К_п$ – коефіцієнт використання потужності; $К_п < 1 = 0,81$.

$К_д$ – коефіцієнт корисної дії, $К_д = 0,78$.

Тоді витрати на силову електроенергію будуть дорівнювати:

$$В_е = \frac{В \cdot П \cdot Ф \cdot К_п}{К_д} = \frac{4,5 \cdot 1,25 \cdot 385 \cdot 0,81}{0,78} = 2248,92 \approx 2249 \text{ грн.}$$

И). Інші витрати $В_{інш}$ можна прийняти як (50…300)% від основної заробітної плати розробників, тобто:

$$В_{інш} = (0,5....3) \times З_о. \quad (4.8)$$

Для нашого випадку отримаємо:

$$В_{інш} = 0,50 \times 41374 = 20687 \text{ грн.}$$

К). Сума всіх попередніх статей витрат складає витрати на виконання магістерської роботи безпосередньо розробником-магістрантом – В.

В = 41374+ 4138 + 10013 + 4201 + 2000 + 2249 + 20687 = 84662 грн.

Л). Загальні витрати на розробку системи автоматизації процесів провайдера безперервного професійного розвитку медичних працівників (БПР) $В_{заг}$ розраховуються за формулою:

$$В_{заг} = \frac{В}{\beta}, \quad (4.9)$$

де β – коефіцієнт, який характеризує етап (стадію) виконання цієї роботи. Можна прийняти, що, $\beta \approx 0,85$ [54].

Тоді: $В_{заг} = \frac{84662}{0,85} = 996023,53$ грн або приблизно 100 тис. грн.

Тобто прогнозовані загальні витрати на розробку нашої системи автоматизації процесів провайдера безперервного професійного розвитку медичних працівників (БПР) становлять приблизно 100 тис. грн.

4.3 Розрахунок економічного ефекту від можливої комерціалізації нашої розробки

Економічний ефект від впровадження та можливої комерціалізації розробленої нами системи автоматизації процесів провайдера безперервного професійного розвитку медичних працівників (БПР) пояснюється її значно кращими функціональними можливостями. Тому нашу розробку можна реалізовувати на ринку дещо дорожче, ніж подібні (але значно гірші) за функціями розробки. Так, якщо подібна за функціями система може коштувати на ринку приблизно 50 тис. грн, то нашу розробку можна буде реалізовувати на ринку приблизно за 60 тис. грн або на 10 тис. грн дорожче.

Аналіз місткості ринку показує, що на сьогодні в Україні кількість потенційних споживачів розробленої нами системи автоматизації процесів провайдера безперервного професійного розвитку медичних працівників (БПР) може становити приблизно 510 осіб. Аналіз ринку також показує, що можна очікувати зростання попиту на нашу розробку принаймні протягом 3-х років після її впровадження.

Тобто, якщо наша розробка буде впроваджена з 1 січня 2025 року, то її результати будуть виявлятися протягом 2025-го, 2026-го та 2027-го років.

Прогноз зростання попиту на нашу розробку складає по роках:

а) 2025 р. – приблизно +20 шт. до базового року;

б) 2026 р. – +30 шт. до базового року;

в) 2027 р. – +40 шт. до базового року.

Можливе збільшення чистого прибутку $\Delta П_i$, що його може отримати потенційний інвестор від комерціалізації нашої розробки, тобто виведення нашої розробки на ринок, становитиме:

$$\Delta П_i = \sum_{1}^{n} (\Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\upsilon}{100}), \qquad (4.10)$$

де $\Delta Ц_о$ – покращення основного якісного показника від впровадження результатів нашої розробки у цьому році. Для нашого випадку це є збільшення ціни реалізації нашої розробки $\Delta Ц_о = (60 – 50) = + 10$ тис. грн;

N – основний кількісний показник, який визначає обсяг діяльності у році до впровадження результатів розробки; N = 510 шт.;

ΔN – покращення основного кількісного показника від впровадження результатів розробки. Таке покращення становитиме по роках, відповідно:

у 2025 році – + 20 шт., у 2026 році +30 шт, та у 2027 році + 40 шт. (відносно базового 2024 року);

$Ц_о$ – основний якісний показник (тобто ціна), який визначає обсяг діяльності у році після впровадження результатів розробки, грн; $Ц_о$= 60 тис. грн;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки; для нашого випадку n = 3;

λ – коефіцієнт, який враховує сплату податку на додану вартість; $\lambda = 0{,}8333$;

ρ – коефіцієнт, який враховує рентабельність продукту. Рекомендується приймати $\rho = (0{,}2...0{,}5)$; візьмемо $\rho = 0{,}4$;

υ – ставка податку на прибуток. У 2024 υ = 18%. У 2025 році і в наступних роках також очікуємо ставку υ = 18%.

Тоді можливе зростання чистого прибутку $\Delta П_1$ для потенційного інвестора протягом першого року від можливого впровадження (комерціалізації) нашої розробки (2025 р.) складе:

$$\Delta П_1 = [10 \cdot 510 + 60 \cdot 20] \cdot 0{,}8333 \cdot 0{,}4 \cdot (1 - \frac{18}{100}) \approx 1722 \text{ тис. грн.}$$

Можливе зростання чистого прибутку $\Delta П_2$ для потенційного інвестора від можливого впровадження (комерціалізації) нашої розробки протягом другого (2026 р.) року складе:

$$\Delta П_2 = [10 \cdot 510 + 60 \cdot 30] \cdot 0{,}8333 \cdot 0{,}4 \cdot (1 - \frac{18}{100}) \approx 1886 \text{ тис. грн.}$$

Можливе зростання чистого прибутку $\Delta П_3$ для потенційного інвестора від можливого впровадження (комерціалізації) нашої розробки протягом третього (2027 р.) року становитиме:

$$\Delta П_3 = [10 \cdot 510 + 60 \cdot 40] \cdot 0,8333 \cdot 0,4 \cdot (1 - \frac{18}{100}) \approx 2050 \text{ тис. грн.}$$

Приведена вартість зростання всіх чистих прибутків, що їх може отримати потенційний інвестор від можливого впровадження нашої розробки становитиме:

$$ПП = \sum_{1}^{т} \frac{\Delta П_i}{(1+\tau)^t}, \quad (4.11)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої роботи, грн;

т – період часу, протягом якого виявляються результати впровадженої роботи, роки. Для нашого випадку т = 3 роки;

τ – ставка дисконтування. Приймемо τ = 0,10 (10%);

t – період часу від моменту початку розроблення системи автоматизації процесів провайдера безперервного професійного розвитку медичних працівників (БПР) до моменту отримання можливих чистих прибутків потенційним інвестором.

Тоді приведена вартість зростання всіх можливих чистих прибутків ПП, що їх може отримати потенційний інвестор від комерціалізації нашої розробки, складе:

$$ПП^{10\%} = \frac{1722}{(1+0,1)^2} + \frac{1886}{(1+0,1)^3} + \frac{2050}{(1+0,1)^4} \approx 1423 + 1417 + 1400 = 4240 \text{ тис. грн.}$$

Теперішня вартість інвестицій PV, що можуть бути вкладені для реалізації нашої розробки: PV = (1,0…5) × $В_{заг}$.

Для нашого випадку приймемо:

PV = (1,0…5) × 100 = 5 × 100 = 500 тис. грн.

Розраховуємо абсолютний ефект від можливих вкладених інвестицій $Е_{абс}$.

$$Е_{абс} = ПП - PV, \quad (4.12)$$

де ПП – приведена вартість збільшення всіх чистих прибутків для потенційного інвестора від можливого впровадження нашої розробки, грн;

PV – теперішня вартість інвестицій PV = 500 тис. грн.

Абсолютний ефект від можливого впровадження нашої розробки (при прогнозованому ринку збуту) за три роки складе:

$$Е_{абс} = 4240 - 500 = 3740 \text{ тис. грн.}$$

Оскільки $Е_{абс} > 0$, то комерціалізація нашої розробки може бути доцільною.

Далі розрахуємо внутрішню дохідність $Е_в$ вкладених інвестицій:

$$Е_в = \sqrt[Т_ж]{1 + \frac{Е_{абс}}{PV}} - 1, \quad (4.13)$$

де $Е_{абс}$ – абсолютний ефект вкладених інвестицій; $Е_{абс}$ = 3740 тис. грн;

PV –теперішня вартість початкових інвестицій PV = 500 тис. грн;

$Т_ж$ – життєвий цикл розробки, роки.

$Т_ж$ = 4 років (2024-й, 2025-й, 2026-й, 2027-й роки)

Для нашого випадку отримаємо:

$$Е_в = \sqrt[4]{1 + \frac{3740}{500}} - 1 = \sqrt[4]{1 + 7,4800} - 1 = \sqrt[4]{8,4800} - 1 = 1,706 - 1 = 0,706 = 70,6\%.$$

Далі визначимо ту мінімальну дохідність, нижче за яку потенційному інвестору не вигідно буде займатися комерціалізацією нашої розробки.

Мінімальна дохідність або мінімальна (бар'єрна) ставка дисконтування $\tau_{мін}$ визначається за формулою:

$$\tau_{мін} = d + f, \quad (4.14)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2024 році в Україні d = (0,14...0,18);

f – показник, що характеризує ризикованість вкладень; f = (0,05...0,40).

Для нашого випадку отримаємо:

$$\tau_{мін} = 0,15 + 0,35 = 0,50 \text{ або } \tau_{мін} = 50\%.$$

Оскільки величина $E_в = 70,6\% > \tau_{мін} = 50\%$, то потенційний інвестор у принципі може бути зацікавлений у фінансуванні та комерціалізації розробленої нами системи автоматизації процесів провайдера безперервного професійного розвитку медичних працівників (БПР).

Далі розраховуємо термін окупності коштів, вкладених у можливу комерціалізацію розробленої системи автоматизації процесів провайдера безперервного професійного розвитку медичних працівників (БПР).

Термін окупності $T_{ок}$ розраховується за формулою:

$$T_{ок} = \frac{1}{E_в}. \qquad (4.15)$$

Для нашого випадку термін окупності $T_{ок}$ коштів становитиме:

$$T_{ок} = \frac{1}{0,706} = 1,42 \text{ років} < 3 \text{ років},$$

що свідчить про потенційну доцільність комерціалізації розробленої нами системи автоматизації процесів провайдера безперервного професійного розвитку медичних працівників (БПР).

Далі проведено моделювання залежності величини внутрішньої дохідності вкладених інвестором потенційних інвестицій від рівня інфляції в країні (який в умовах війни може зростати).

Прийнявши рівень інфляції у 20%, отримаємо:

$$ПП^{20\%} = \frac{1722}{(1+0,2)^2} + \frac{1886}{(1+0,2)^3} + \frac{2050}{(1+0,2)^4} \approx 1195 + 1091 + 989 = 3275 \text{ тис. грн.}$$

Тоді абсолютний ефект від можливого впровадження нашої розробки за три роки складе:

$$E_{абс} = 3275 - 500 = 2775 \text{ тис. грн.}$$

Внутрішня дохідність $E_в$ вкладених інвестицій становитиме:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1,$$

де $E_{абс}$ – абсолютний ефект вкладених інвестицій; $E_{абс} = 2775$ тис. грн;

PV –теперішня вартість початкових інвестицій PV = 500 тис. грн.

Для нашого випадку отримаємо:

$$E_в = \sqrt[4]{1+\frac{2775}{500}} - 1 = \sqrt[4]{1+5,5500} - 1 = \sqrt[4]{6,5500} - 1 = 1,6 - 1 = 0,6 = 60,0\%.$$

Прийнявши рівень інфляції у 30%, отримаємо:

$$ПП^{30\%} = \frac{1722}{(1+0,3)^2} + \frac{1886}{(1+0,3)^3} + \frac{2050}{(1+0,3)^4} \approx 1019 + 859 + 718 = 2596 \text{ тис. грн.}$$

Тоді абсолютний ефект від можливого впровадження нашої розробки за три роки складе:

$$E_{абс} = 2596 - 500 = 2096 \text{ тис. грн.}$$

Внутрішня дохідність $E_в$ вкладених інвестицій становитиме:

$$E_в = \sqrt[T_ж]{1+\frac{E_{абс}}{PV}} - 1,$$

де $E_{абс}$ – абсолютний ефект вкладених інвестицій; $E_{абс}$ = 2096 тис. грн;

PV –теперішня вартість початкових інвестицій PV = 500 тис. грн.

Для нашого випадку отримаємо:

$$E_в = \sqrt[4]{1+\frac{2096}{500}} - 1 = \sqrt[4]{1+4,1920} - 1 = \sqrt[4]{5,1920} - 1 = 1,51 - 1 = 0,51 = 51,0\%.$$

Зроблені розрахунки у вигляді графіків наведено на рис. 4.1.

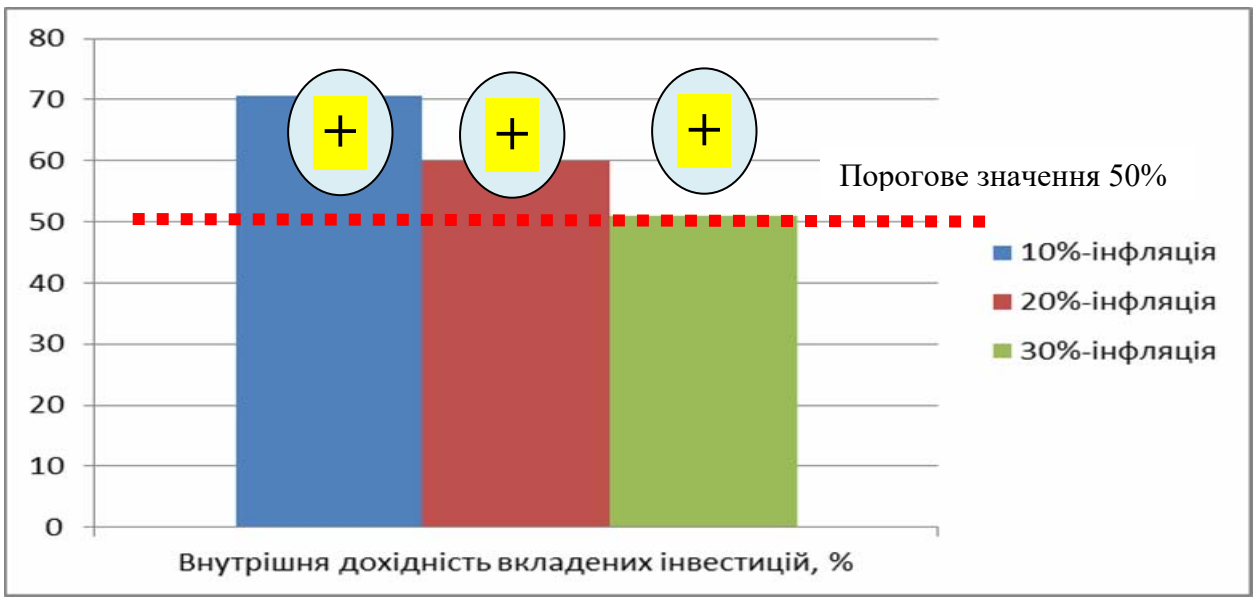


Рисунок 4.1 – Моделювання залежності величини внутрішньої дохідності потенційних інвестицій, вкладених в комерціалізацію розробленої нами системи автоматизації процесів провайдера БПР, від рівня інфляції в країні

Аналіз графіка на рис 5.1 показує, що при рівні інфляції в країні до 30% величина внутрішньої дохідності інвестицій буде перевищувати порогове значення $\tau_{мін}$ = 50%, і тому комерціалізація нашої розробки може бути доцільною.

Це ще раз підкреслює високий технічний рівень нашої розробки та її високу актуальність і затребуваність.

4.4 Висновки до розділу 4

Результати виконаної економічної частини магістерської кваліфікаційної роботи зведено у таблицю:

Показники	Задані у ТЗ	Досягнуті у магістерській кваліфікаційній роботі	Висновок
1. Витрати на розробку	Приблизно 100 тис. грн	100 тис. грн.	Практично досягнуто
2. Абсолютний ефект від впровадження розробки, тис. грн	Не менше 3500 тис. грн (за три роки)	3740 тис. грн	Практично виконано
3. Внутрішня дохідність потенційних інвестицій, %	не менше 50%	70,6%	Досягнуто
4. Термін окупності потенційних інвестицій, роки	до 3-ти років	1,42 років	Виконано

Таким чином, основні техніко-економічні показники розробленої нами системи автоматизації процесів провайдера безперервного професійного розвитку медичних працівників (БПР), визначені у технічному завданні, виконані.

ВИСНОВКИ

В першому розділі встановлено, що на сучасному світовому ринку не представлено рішень для автоматизації процесів провайдера безперервного професійного розвитку медичних працівників, оскільки система БПР є специфічною для України. Також, на українському ринку також не представлено opensource рішень для автоматизації процесів провайдера безперервного професійного розвитку медичних працівників. Деякі варіанти рішень були розроблені під замовлення великих провайдерів для застосування у власній мережі. Для малих або середніх провайдерів подібних аналогів на українському ринку немає, а дозволити собі розробку або покупку рішень, які розраховані на великих провайдерів вони не можуть, оскільки це досить дорого і використовувати такі рішення невигідно з боку бюджету таких провайдерів.

Відзначено, що буде актуальним та доцільним розробка бюджетного цілісного рішення, яке задовольнить потреби в автоматизації процесів провайдера безперервного професійного розвитку медичних працівників.

У другому розділі виконано аналіз різних типів архітектур систем, таких як монолітна, клієнт-серверна, розподілена та мікросервісна архітектура. Проведено порівняльний аналіз переваг та недоліків кожного типу з метою визначення оптимального варіанту для розробки системи.

В результаті проведеного аналізу ми обрали монолітну архітектуру. Це обумовлено тим, що додаток має працювати у кожного провайдера окремо і не буде суттєво навантажений. Водночас, відмовилися від клієнт-серверної, розподіленої та мікросервісної архітектур через їхні недоліки, такі як складність розробки та розгортання, ускладнене управління додатком і підвищений ризик випадкових збоїв через розділений характер системи.

Також по результатах проведеного аналізу ми обрали PHP для розробки монолітної архітектури, що обумовлено його простотою, продуктивністю, широкою підтримкою спільноти та індустрії, а також

наявністю багатої екосистеми інструментів та фреймворків, що робить його привабливим та ефективним вибором для створення нашого веб-додатку.

Вибір між SQL та NoSQL базами даних залежить від конкретних вимог проекту. SQL бази даних зазвичай краще підходять для додатків з жорсткою структурою даних, що вимагають надійності та ACID-сумісності, у той час як NoSQL бази даних частіше використовуються для додатків з великими обсягами даних, де важлива гнучкість та масштабованість. Загалом застосування SQL баз даних у веб-додатку забезпечує надійне, структуроване та ефективне зберігання даних, що є ключовим компонентом для успішного функціонування додатку та забезпечення задоволення потреб користувачів.

Серед SQL баз даних ми обрали застосування MySQL баз даних у розробці, що має забезпечити надійне, продуктивне та легко масштабоване сховище даних для програм будь-якого масштабу. Її простота використання, широка підтримка та висока продуктивність зробили її популярним та привабливим вибором для нашого веб-додатку.

У даному проекті віддається перевага ручному кодуванню інтерфейсу користувача без застосування готових конструкторів, оскільки розробник має достатній досвід для подолання можливих труднощів, що дозволяє зберегти контроль над всіма аспектами розробки.

Також аналіз продемонстрував, що для простих проектів часто краще обходитися без використання фреймворків, оскільки це дозволяє зберегти простоту, легкість та продуктивність коду. Чистого HTML, CSS та JavaScript цілком достатньо для створення невеликих та простих веб-додатків. Тому при розробці інтерфейсу в даному проекті буде віддана перевага застосування саме чистого HTML, CSS та JavaScript.

У третьому розділі був проведений детальний аналіз та опис основних етапів розробки системи автоматизації процесів провайдера безперервного професійного розвитку медичних працівників.

Детально розглянуто структуру бази даних системи, яка складається з наступних таблиць: "courses", "students", "test_bases", "test_questions",

"test_try", "test_answers", "instructors", "course_type", "course_format", "spec". Кожна таблиця відіграє ключову роль у функціонуванні системи та зберіганні відповідної інформації.

Використання PHP є ключовим у створенні системи, що дозволяє реалізувати необхідні компоненти з урахуванням архітектурних вимог. Це забезпечує гладку взаємодію між системою та клієнтом. Система відповідає за обробку даних, отриманих від клієнта, та виконання запитів до бази даних.

Детально розглянуто архітектуру системи, яка розподілена на окремі папки – "includes", "templates" та "vendor", які розміщено в кореневій папці. Також в кореневій папці розміщені основні індексні файли – "index.php" та "admin.php", файли з налаштуваннями composer – "composer.json" та "composer.lock", а також файл з конфігурацією web-сервера apache – ".htaccess".

В економічному розділі була розглянута доцільність комерціалізації розробки системи автоматизації процесів провайдера безперервного професійного розвитку медичних працівників. Основні техніко-економічні показники (характеристики) розробленого програмного забезпечення визначені у технічному завданні, виконані.

Результати роботи були впроваджені в роботу діючого провайдера безперервного професійного розвитку медичних працівників Товариство з обмеженою відповідальністю «Агенція безперервного професійного розвитку». Акт впровадження № 1 від 04.06.2024, продемонстровано в додатку Г.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Постанова Кабінету Міністрів України від 14.07.2020 р. №725 «Про затвердження Положення про систему безперервного професійного розвитку медичних та фармацевтичних працівників». URL: https://zakon.rada.gov.ua/laws/show/725-2021-%D0%BF#Text (Дата звернення: 05.03.2024)

2. Наказ МОЗ України від 12.08.2009 р. №588 «Про атестацію професіоналів з вищою немедичною освітою, які працюють в системі охорони здоров'я». URL: https://zakon.rada.gov.ua/laws/show/z0895-09#Text (Дата звернення: 05.03.2024)

3. Система БПР станом на 2022-2024 роки. URL: https://umaedu.com.ua/navchannya/bezperervnyj-profesijnyj-rozvytok/ (Дата звернення: 05.03.2024)

4. Система БПР працівників сфери охорони здоров’я: зміни і перспективи розвитку. URL: https://seminar.expertus.com.ua/static/docs/362_Худякова.pdf (Дата звернення: 05.03.2024)

5. Наказ МОЗ України від 22.02.2019 р. №446 «Деякі питання безперервного професійного розвитку лікарів». URL: https://zakon.rada.gov.ua/laws/show/z0293-19#Text (Дата звернення: 05.03.2024)

6. Безперервний професійний розвиток медиків - 2024. URL: https://medplatforma.com.ua/article/1986-bezperervniy-rozvitok-lkarv-yak-nabrati-bali (Дата звернення: 05.03.2024)

7. Проєкт наказу МОЗ України від 3.11.2023 р. "Про затвердження Порядку функціонування електронної системи забезпечення безперервного професійного розвитку працівників сфери охорони здоров’я". URL: https://moz.gov.ua/article/public-discussions/proekt-nakazu-moz-ukraini-pro-zatverdzhennja-porjadku-funkcionuvannja-elektronnoi-sistemi-zabezpechennja-

bezperervnogo-profesijnogo-rozvitku-pracivnikiv-sferi-ohoroni-zdorov%e2%80%99ja (Дата звернення: 05.03.2024)

8. Центр тестування при МОЗ України - еБПР. URL: https://www.testcentr.org.ua/uk/ebpr (Дата звернення: 05.03.2024)

9. Провайдери заходів БПР. URL: https://docs.google.com/spreadsheets/d/1F4sPSsTIApG5unDSIu5y6wIyxNnGA7bq/edit#gid=1089264369

10. Перелік заходів БПР 2024. URL: https://docs.google.com/spreadsheets/d/1QZJ57A0HPzzW_-9c3JYHeMzeuS08Wj5G4_eVgoQuf8Y/edit#gid=2117408314

11. Бали БПР 2024. URL: https://docs.google.com/spreadsheets/d/1l21I0aaz61tduXAJYcwuSW_PJBDz7n9M/edit#gid=914527139

12. Центр тестування при МОЗ України - БПР. URL: https://www.testcentr.org.ua/uk/bpr (Дата звернення: 05.03.2024)

13. Агенція екстреної медичної допомоги - БПР - Медичні / Фармацевтичні спеціальності. URL: https://aemc.org.ua/bpr/spec/ (Дата звернення: 05.03.2024)

14. Реєстрація заходів БПР на 2024 рік. URL: https://forms.gle/7u6ZDvFwCNSN3aJ8A (Дата звернення: 05.03.2024)

15. Облік балів безперервного професійного розвитку за заходами 2024 року. URL: https://forms.gle/iMFBVRhYAEHxQW5R9 (Дата звернення: 05.03.2024)

16. Dental.ua - Громадська Спілка Безперервного Професійного Розвитку Стоматологів. URL: https://dental.ua/ (Дата звернення: 05.03.2024)

17. Medvoice – Міжнародний медичний науково-освітній сервіс. URL: https://medvoice.net/ (Дата звернення: 05.03.2024)

18. Професійна платформа | Навчання лікарів | бали БПР. URL: https://pro.docos.one/ (Дата звернення: 05.03.2024)

19. Магазин Progress. URL: https://progressplatform.org/ (Дата звернення: 05.03.2024)

20. Провайдер БПР №1043: семінари та конференції для лікарів. URL: https://med-expert.com.ua/ (Дата звернення: 05.03.2024)

21. Агенція екстреної медичної допомоги - Головна. URL: https://aemc.org.ua/ (Дата звернення: 05.03.2024)

22. ЛЕКЦІЯ 1-1. Архітектура програмного забезпечення. https://org2.knuba.edu.ua/mod/resource/view.php?id=9951 (Дата звернення: 15.03.2024)

23. Монолітна архітектура ПЗ. URL: https://qalight.ua/baza-znaniy/shho-take-monolitna-arhitektura/ (Дата звернення: 15.03.2024)

24. Що краще моноліт чи мікросервіси? URL: https://iampm.club/ua/blog/shho-krashhe-monolit-chi-mikroservisi-yak-obrati-arhitekturu-projektu/ (Дата звернення: 15.03.2024)

25. Сікайло В. О., Кравченко С. М. Необхідність проектування гнучкого дизайну для монолітної архітектури. URL: https://conf.ztu.edu.ua/wp-content/uploads/2023/02/53.pdf (Дата звернення: 15.03.2024)

26. Архітектура клієнт-сервер - Електронна бібліотека. URL: http://inter.ptngu.com/kompyuterni-merezhi/arhitektura-kliyent-server (Дата звернення: 15.03.2024)

27. Балик Н.Р., Мандзюк В.І. Сучасні клієнт-серверні технології та їх застосування при вивченні систем управління базами даних. URL: https://fi.npu.edu.ua/files/Zbirnik_KOSN/12/29.pdf (Дата звернення: 15.03.2024)

28. Савченко О.С. Архітектура розподіленої багаторівневої системи виявлення шкідливого програмного забезпечення в локальних комп’ютерних мережах // Вчені записки ТНУ імені В.І. Вернадського. Серія: технічні науки. Том 29 (68). № 2. 2018. URL: https://tech.vernadskyjournals.in.ua/journals/2018/2_2018/32.pdf (Дата звернення: 15.03.2024)

29. Архітектура розподілених систем. URL: http://kimt.uad.edu.ua/arkhitektura-rozpodilenikh-sistem-fvpit-5.html (Дата звернення: 15.03.2024)

30. Мікросервісна архітектура ПЗ. URL: https://qalight.ua/baza-znaniy/shho-take-mikroservisna-arhitektura-pz/ (Дата звернення: 15.03.2024)

31. Мікросервісна архітектура: основні концепції та виклики. URL: https://foxminded.ua/mikroservisna-arkhitektura/ (Дата звернення: 15.03.2024)

32. Що таке веб-сервер? – SERVER SOLUTIONS. https://serversolutions.com.ua/blogs/news/%D1%89%D0%BE-%D1%82%D0%B0%D0%BA%D0%B5-%D0%B2%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80 (Дата звернення: 24.03.2024)

33. Java. URL: https://www.java.com/ru/ (Дата звернення: 24.03.2024)

34. Java - історія створення, сфера застосування, переваги та недоліки. URL: https://muff.kiev.ua/content/java-istoriya-sozdaniya-sfera-primeneniya-dostoinstva-i-nedostatki (Дата звернення: 24.03.2024)

35. PHP: Hypertext Preprocessor. URL: https://www.php.net/ (Дата звернення: 24.03.2024)

36. Чому мові PHP роками пророкують «смерть», але вона досі актуальна? URL: https://robotdreams.cc/uk/blog/436-chomu-movi-php-rokami-prorokuyut-smert-ale-vona-dosi-aktualna (Дата звернення: 24.03.2024)

37. JavaScript - MDN Web Docs - Mozilla. URL: https://developer.mozilla.org/ru/docs/Web/JavaScript (Дата звернення: 24.03.2024)

38. Node.js — Run JavaScript Everywhere. URL: https://nodejs.org/ (Дата звернення: 24.03.2024)

39. Фреймворк Node.js — ідеальний інструмент для створення мережевих додатків. URL: https://gl.ua/pl/blog/freymvork-nodejs-idealnyy-instrument-dlya-stvorennya-merezhevykh (Дата звернення: 24.03.2024)

40. Welcome to Python.org. URL: https://www.python.org/ (Дата звернення: 24.03.2024)

41. Популярність мови програмування Python: причини та переваги. URL: https://buki.com.ua/blogs/populiarnist-movi-programuvannia-python-pricini-ta-perevagi/ (Дата звернення: 24.03.2024)

42. Швець М.Ю., Заруба Д.С., Хохлов Ю.В. Порівняння SQL та NoSQL баз даних // Вчені записки ТНУ імені В.І. Вернадського. Серія: технічні науки. Том 29 (68). Ч. 2. № 6. 2018. URL: https://tech.vernadskyjournals.in.ua/journals/2018/6_2018/part_2/7.pdf (Дата звернення: 27.03.2024)

43. Порівняльний аналіз баз даних SQL та NOSQL. URL: https://jvestnik-sss.donnu.edu.ua/article/view/14717 (Дата звернення: 27.03.2024)

44. MySQL. URL: https://www.mysql.com/ (Дата звернення: 27.03.2024)

45. Мамалига Н.Є., Катаєва А.І. Система керування базами даних в сучасних умовах IT-індустрії. URL: https://jvestnik-sss.donnu.edu.ua/article/view/10004/9931 (Дата звернення: 27.03.2024)

46. PostgreSQL: The world's most advanced open source database. URL: https://www.postgresql.org/ (Дата звернення: 27.03.2024)

47. Переваги та Недоліки PostgreSQL. URL: https://prezi.com/p/k9j0zsyzstgc/postgresql/ (Дата звернення: 27.03.2024)

48. Битва титанів: що краще — PostgreSQL чи MySQL? URL: https://highload.today/uk/postgresql-vs-mysql/ (Дата звернення: 27.03.2024)

49. Пастернак І.І. Принципи побудови інтерфейсу користувача кіберфізичної системи // Ком’пютерні системи та мережі. Т. 1. № 1. 2019. URL: https://science.lpnu.ua/sites/default/files/journal-paper/2020/feb/21047/var1ksm-19-55-64.pdf (Дата звернення: 27.03.2024)

50. Мова HTML - що це таке, визначення та поняття. URL: https://uk.economy-pedia.com/11036008-html-language (Дата звернення: 27.03.2024)

51. HTML і CSS довідник українською. URL: https://html-css.co.ua/ (Дата звернення: 27.03.2024)

52. Болотіна В. В., Огляд популярних Javascript фреймворків // Житомирський державний технологічний університет. Секція 2. Інформаційні технології та кібербезпека. URL: https://conf.ztu.edu.ua/wp-content/uploads/2019/06/42.pdf (Дата звернення: 27.03.2024)

53. Чому важливо вивчати Javascript, а не фреймворки? URL: https://web-developer.in.ua/assets/articles/js/learn-js/learn-js.html (Дата звернення: 27.03.2024)

54. Козловський В.О., Лесько О.Й., Кавецький В.В. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт. Вінниця: ВНТУ, 2021. – 42 с.

55. Чорнокнижний С.І. Обґрунтування необхідності розробки системи автоматизації процесів провайдера безперервного професійного розвитку медичних працівників. ВНТКП ВНТУ. Факультет інтелектуальних інформаційних технологій та автоматизації. LIII Всеукраїнська науково-технічна конференція факультету інтелектуальних інформаційних технологій та автоматизації (2024). URL: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2024/paper/view/20635/17075 (Дата звернення: 25.04.2024)

ДОДАТКИ

Додаток А (обов'язковий)

Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інтелектуальних інформаційних технологій та автоматизації

ЗАТВЕРДЖУЮ

Завідувач кафедри АІІТ

д.т.н., проф. Олег БІСІКАЛО

«___» _____ 202__ року

ТЕХНІЧНЕ ЗАВДАННЯ

на магістерську кваліфікаційну роботу

СИСТЕМА АВТОМАТИЗАЦІЇ ПРОЦЕСІВ ПРОВАЙДЕРА БЕЗПЕРЕРВНОГО ПРОФЕСІЙНОГО РОЗВИТКУ МЕДИЧНИХ ПРАЦІВНИКІВ

08-31.МКР.010.02.000 ТЗ

Керівник: к.т.н., проф. каф. АІІТ

_____ Євген ПАЛАМАРЧУК

«___» _____ 202__ р.

Розробив студент гр. АКІТ-22мз

_____ Сергій ЧОРНОКНИЖНИЙ

«___» _____ 202__ р.

Вінниця 2024

1. Назва та галузь застосування.

Система автоматизації процесів провайдера безперервного професійного розвитку медичних працівників. Галузь застосування: Інформаційні системи та технології.

2. Підстава для проведення робіт.

Підставою для виконання роботи є наказ №___ по ВНТУ від «___» _____ 202__ р., та індивідуальне завдання на МКР, затверджене протоколом №___ засідання кафедри АІІТ від «___» _____ 202__ р.

3. Мета та призначення роботи.

Метою даної роботи є розробка системи автоматизації процесів провайдера безперервного професійного розвитку медичних працівників.

4. Джерела розробки:

4.1 PHP: Hypertext Preprocessor. URL: https://www.php.net/ (Дата звернення: 24.03.2024)

4.2 MySQL. URL: https://www.mysql.com/ (Дата звернення: 27.03.2024).

4.3 HTML і CSS довідник українською. URL: https://html-css.co.ua/ (Дата звернення: 27.03.2024)

5. Показники призначення

5.1 Мінімальні вимоги до техніки:

— Процесор: Intel Core i5 3 GHz або вище;

— Об’єм оперативної пам’яті: 8 Gb або більше;

— Жорсткий диск: 120 Gb SATA2 або вище.

5.2 Технології для серверної частини системи – PHP.

5.3 Технології для клієнтської частини системи – HTML і CSS.

5.4 База даних – MySQL.

6. Економічні показники

До економічних показників входять:

— Витрати на розробку – не більше 100 тис. грн;

— Абсолютний ефект від впровадження розробки – не менше 3500 тис. грн;

— Внутрішня дохідність інвестицій – не менше 50%;

— Термін окупності інвестицій – до 3-х років.

7. Стадії розробки:

7.1 Аналіз діяльності та процесів провайдерів БПР - 12.03 - 31.03.24

7.2 Аналіз технологій для розробки системи автоматизації процесів провайдера БПР - 01.04 - 14.04.24

7.3 Програмна реалізація системи автоматизації процесів провайдера БПР - 15.04 - 30.04.24

7.4 Підготовка економічного розділу - 01.05 - 14.05.24

7.5 Оформлення пояснювальної записки і графічного матеріалу - 15.05 - 29.05.24

8. Порядок контролю та приймання

8.1 Рубіжний контроль провести до 29.05.2024.

8.2 Попередній захист МКР провести до 30.05.2024.

8.3 Захист МКР провести до 20.06.2024.

Розробив студент

групи АКІТ-22мз _____ Сергій ЧОРНОКНИЖНИЙ

Додаток Б (обов’язковий)

Ілюстративна частина

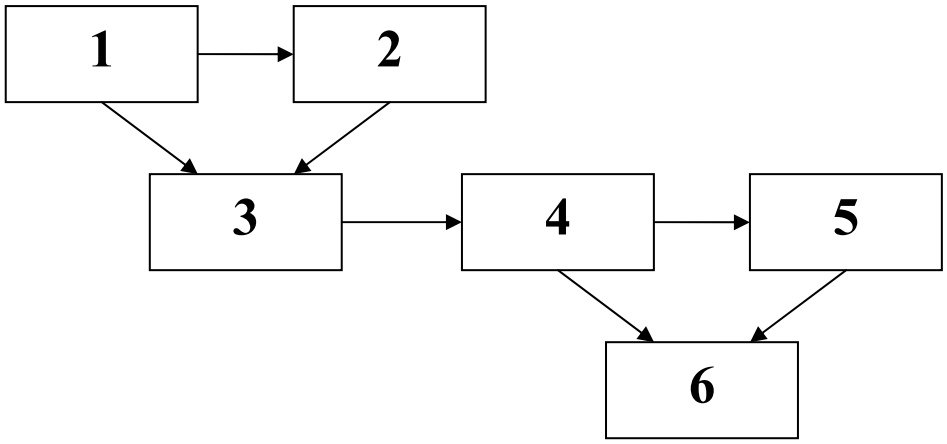


Рисунок Б.1 – Взаємозв’язок процесів, які виконує провайдер під час організації та проведення заходу БПР

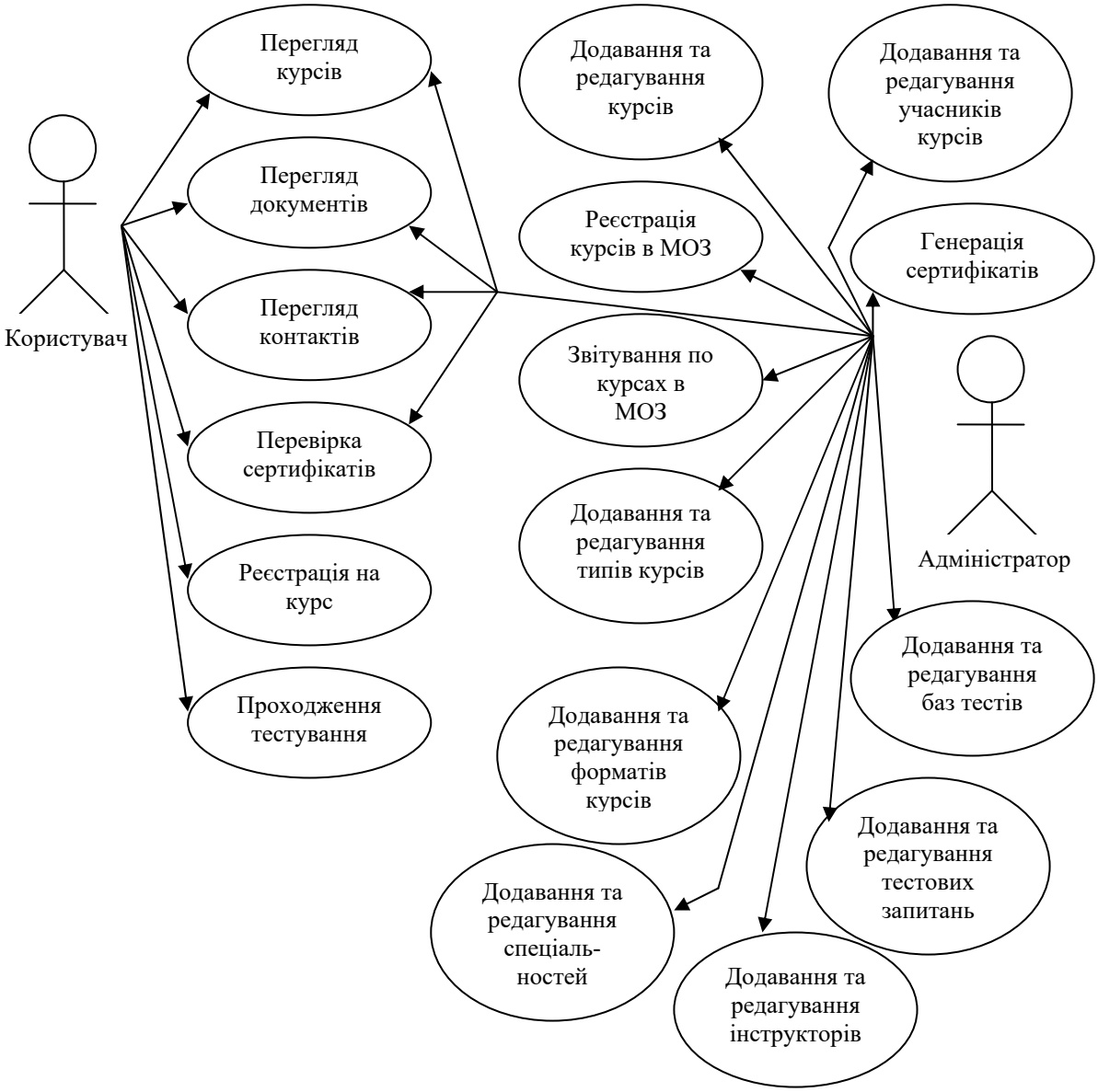


Рисунок Б.2 – Діаграма випадків використання

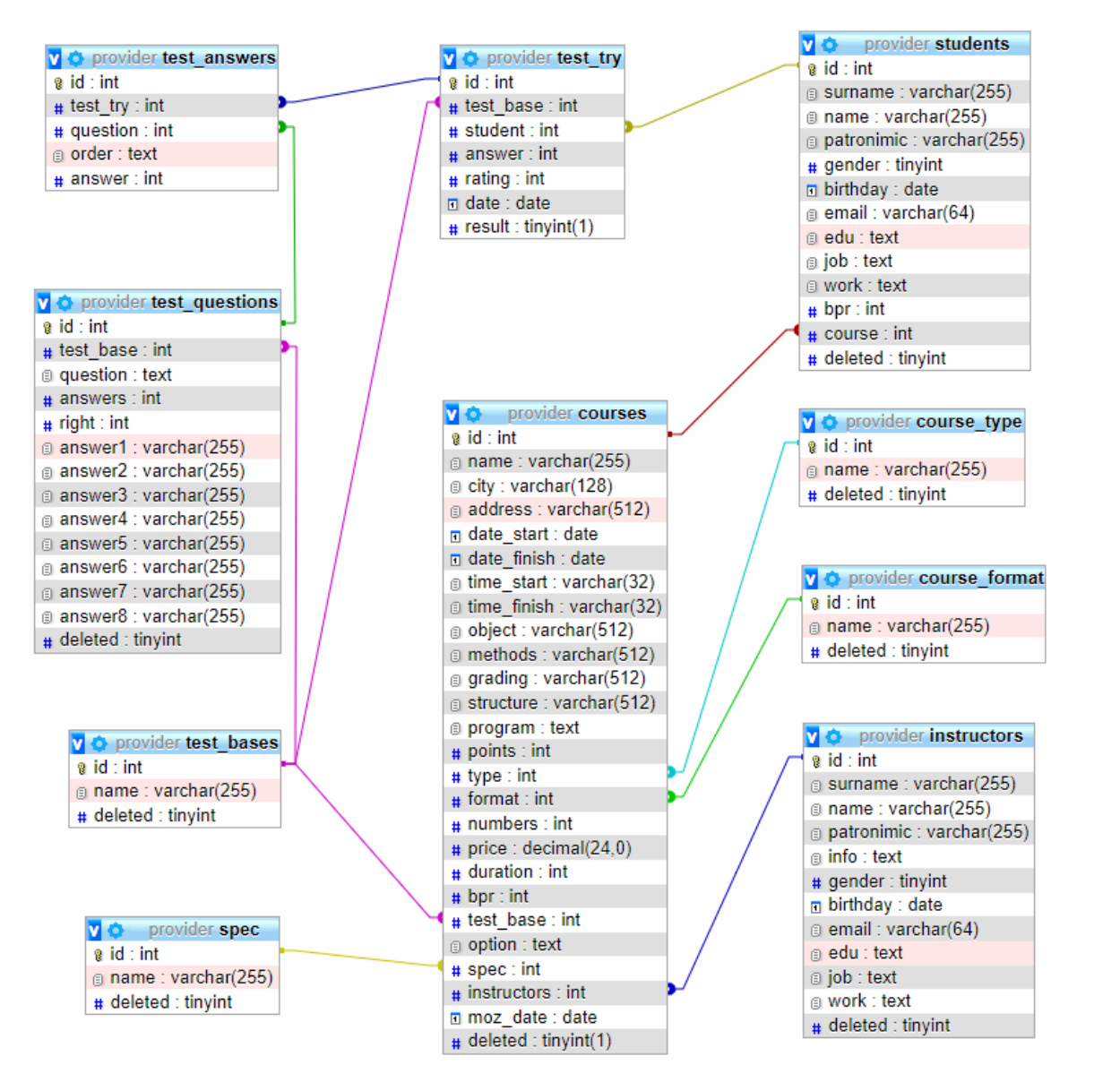


Рисунок 3.3 – Схема структури бази даних

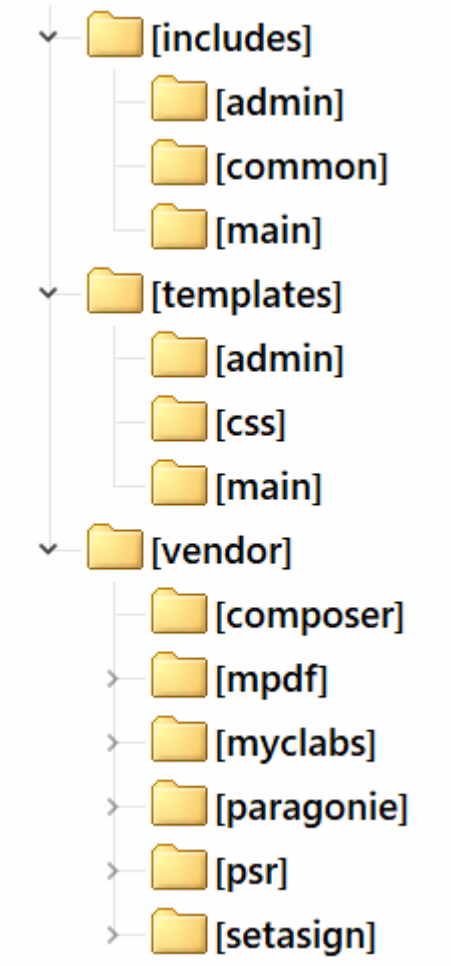


Рисунок Б.4 – Структура каталогів системи

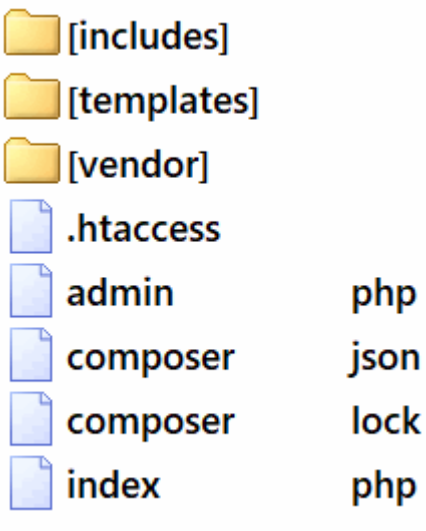


Рисунок Б.5 – Список файлів та папок в кореневій папці системи

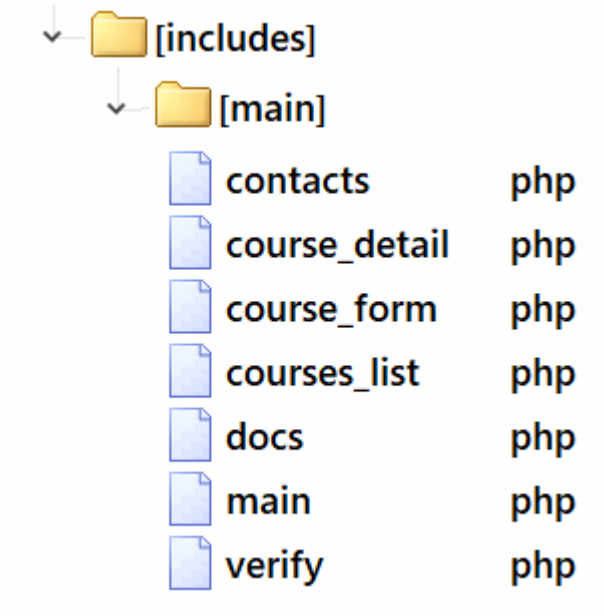


Рисунок Б.6 – Список файлів в папці "includes/main"

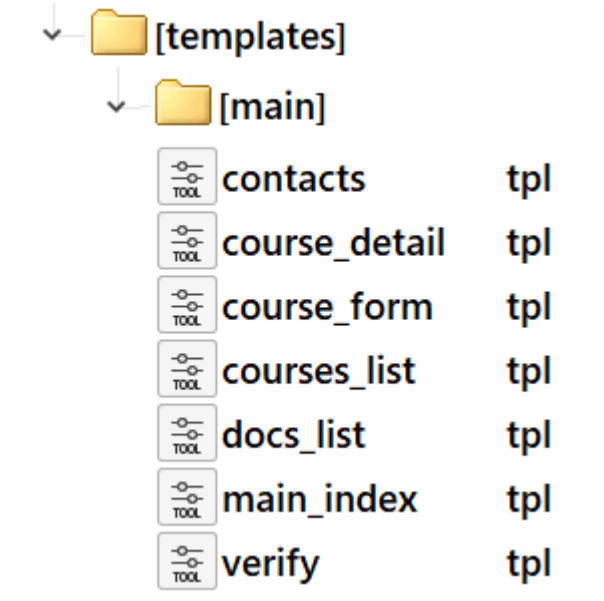


Рисунок Б.7 – Список файлів в папці "templates/main"

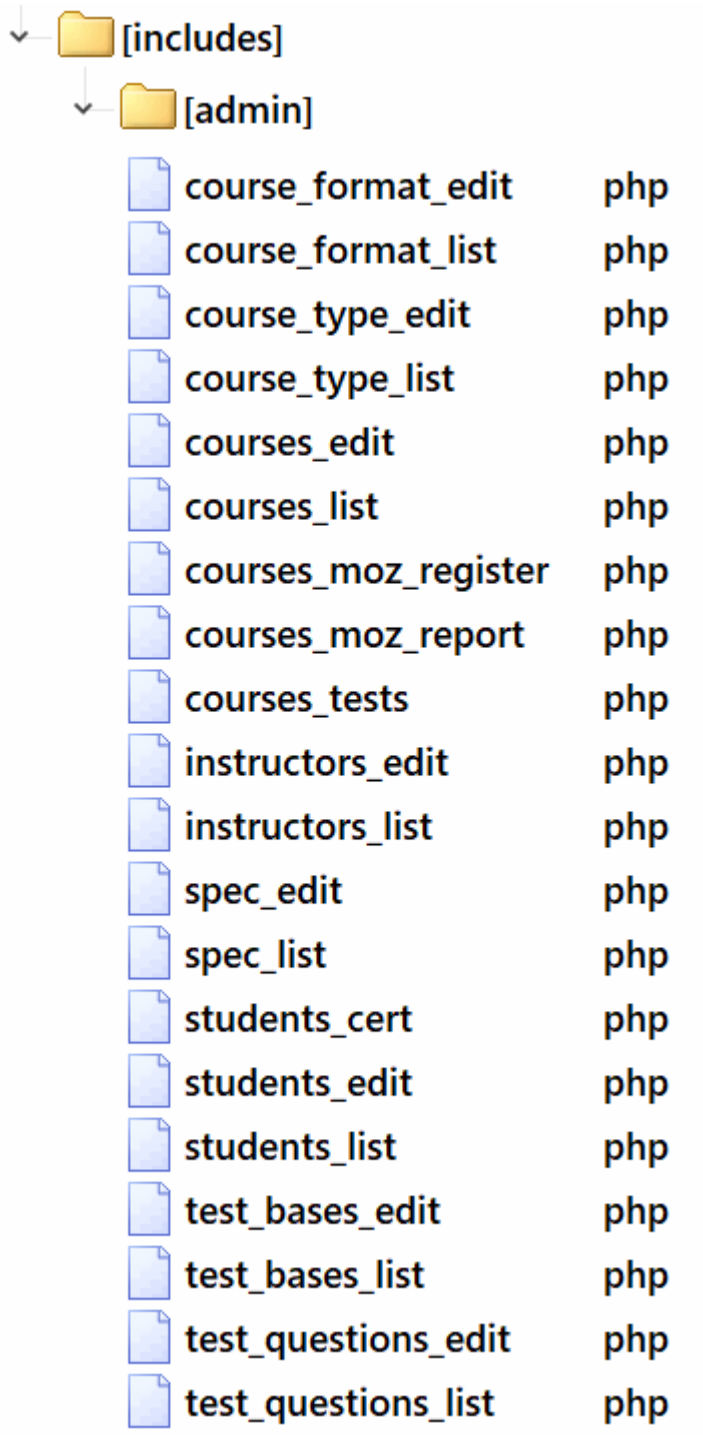


Рисунок Б.8 – Список файлів в папці "includes/admin"

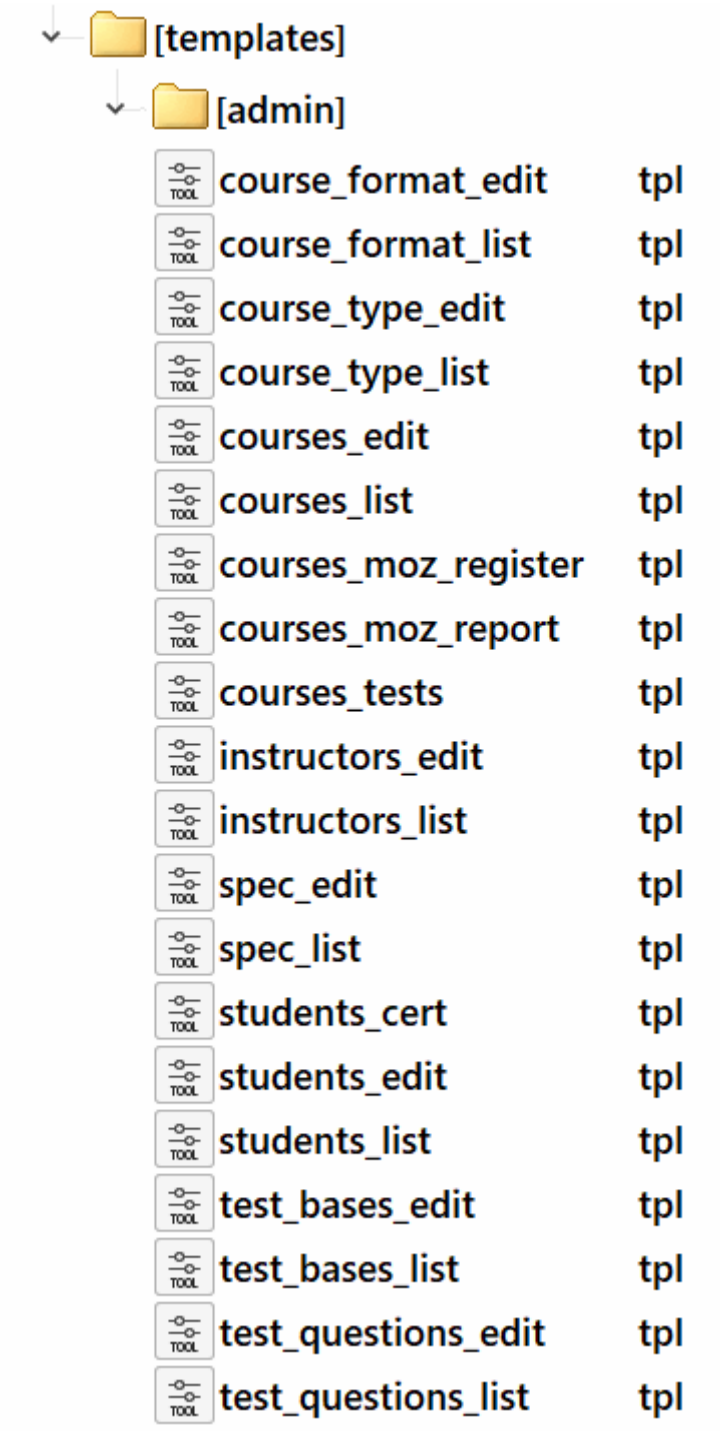


Рисунок Б.9 – Список файлів в папці "templates/admin"

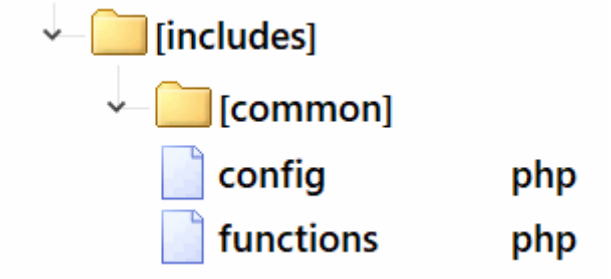


Рисунок Б.10 – Список файлів в папці "includes/common"

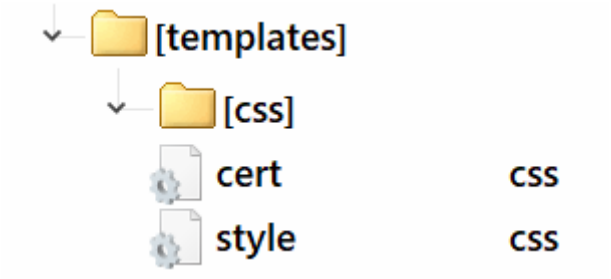


Рисунок Б.11 – Список файлів в папці "templates/css"

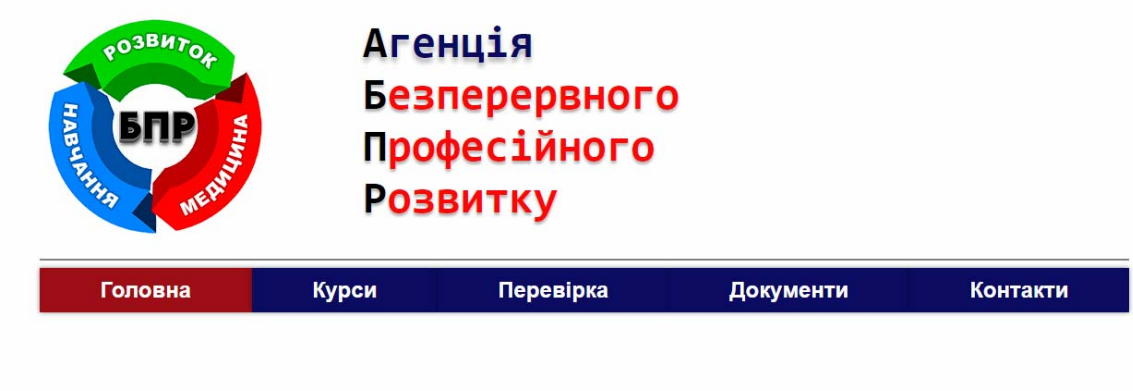


Рисунок Б.12 – Інтерфейс головної сторінки

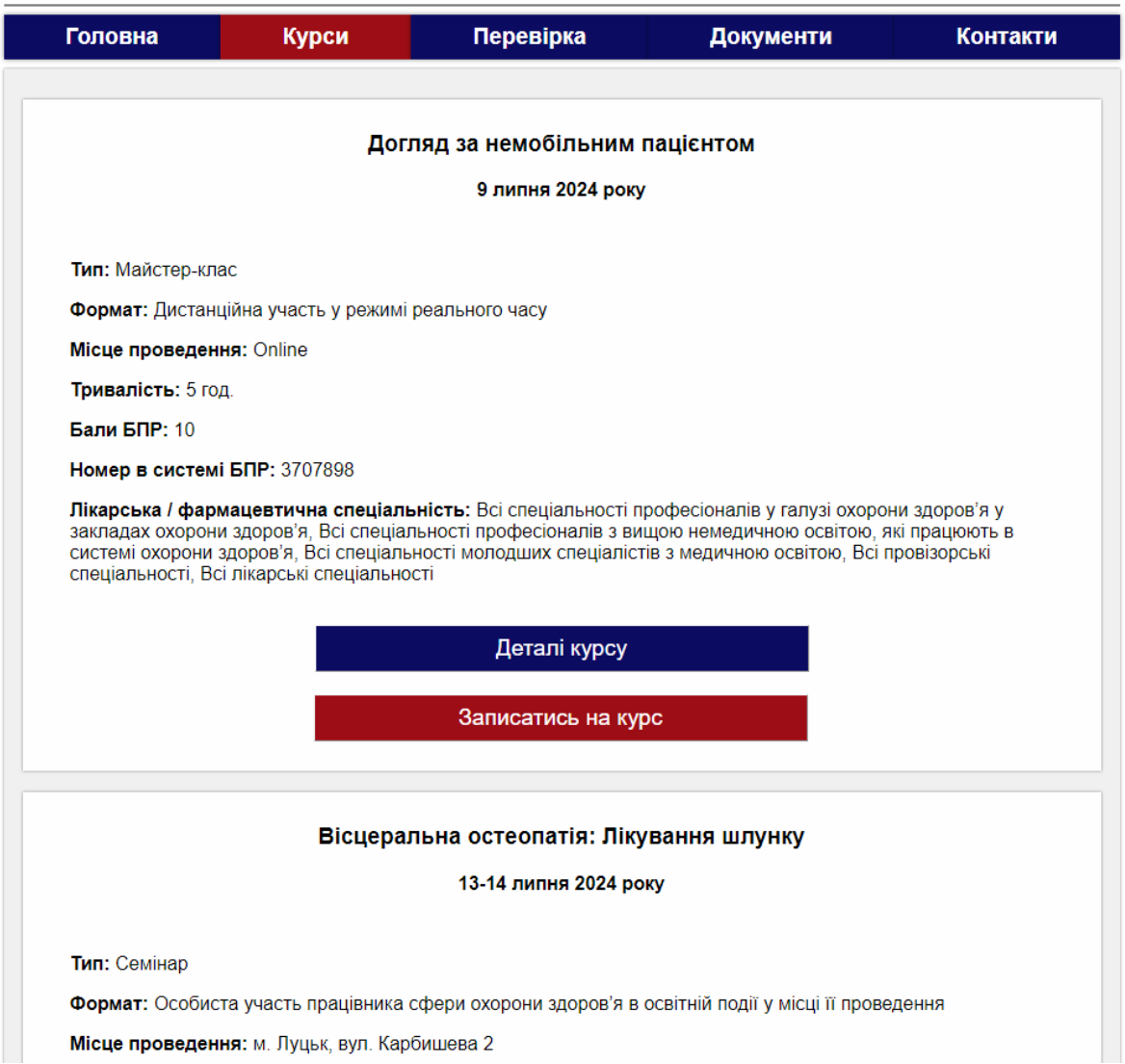


Рисунок Б.13 – Інтерфейс сторінки "Курси"

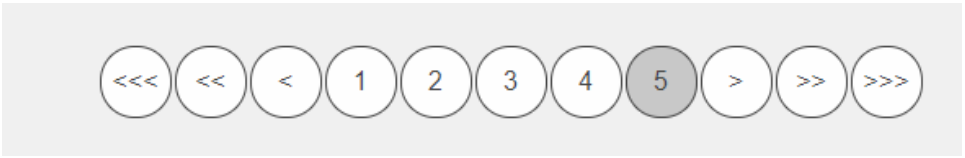
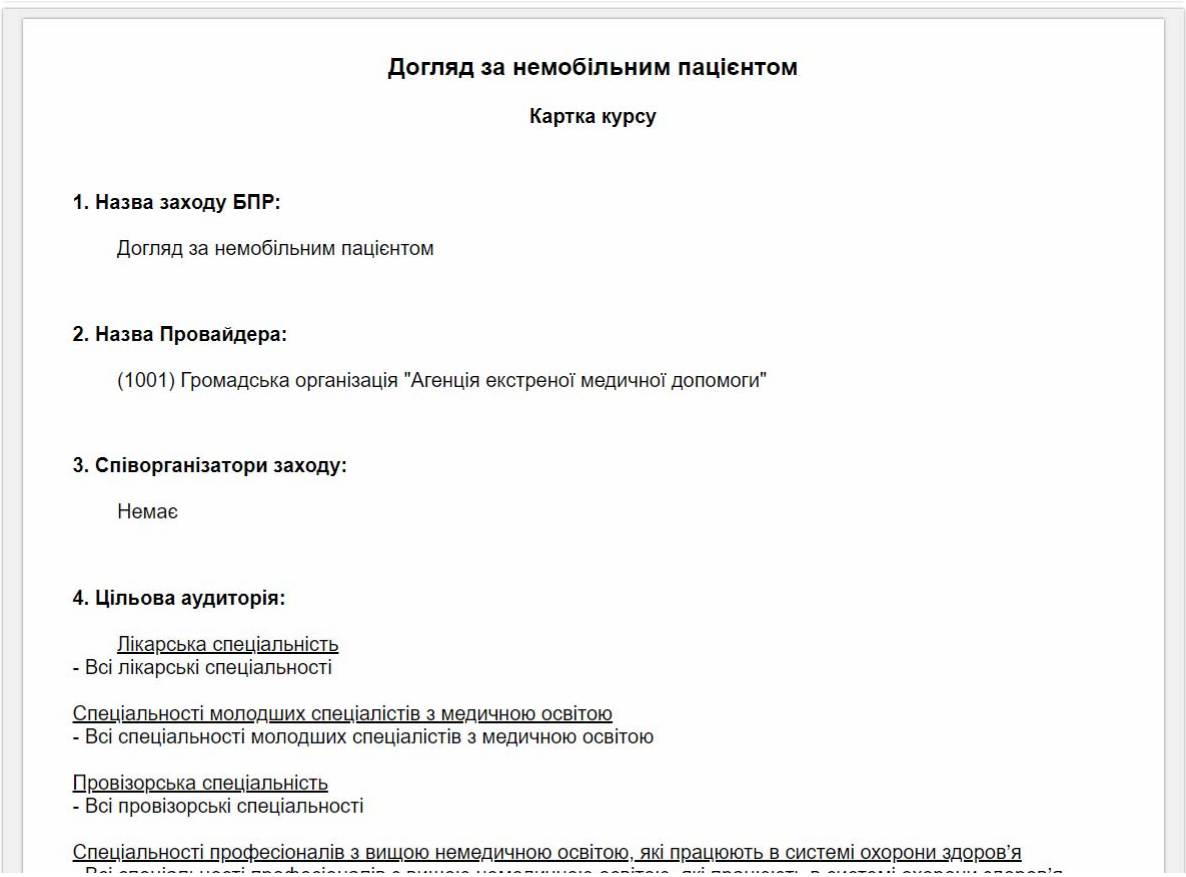


Рисунок Б.14 – Інтерфейс механізму розділення на сторінки



Догляд за немобільним пацієнтом

Картка курсу

1. Назва заходу БПР:

Догляд за немобільним пацієнтом

2. Назва Провайдера:

(1001) Громадська організація "Агенція екстреної медичної допомоги"

3. Співорганізатори заходу:

Немає

4. Цільова аудиторія:

Лікарська спеціальність
- Всі лікарські спеціальності

Спеціальності молодших спеціалістів з медичною освітою
- Всі спеціальності молодших спеціалістів з медичною освітою

Провізорська спеціальність
- Всі провізорські спеціальності

Спеціальності професіоналів з вищою немедичною освітою, які працюють в системі охорони здоров'я

Рисунок Б.15 – Інтерфейс сторінки "Деталі курсу"

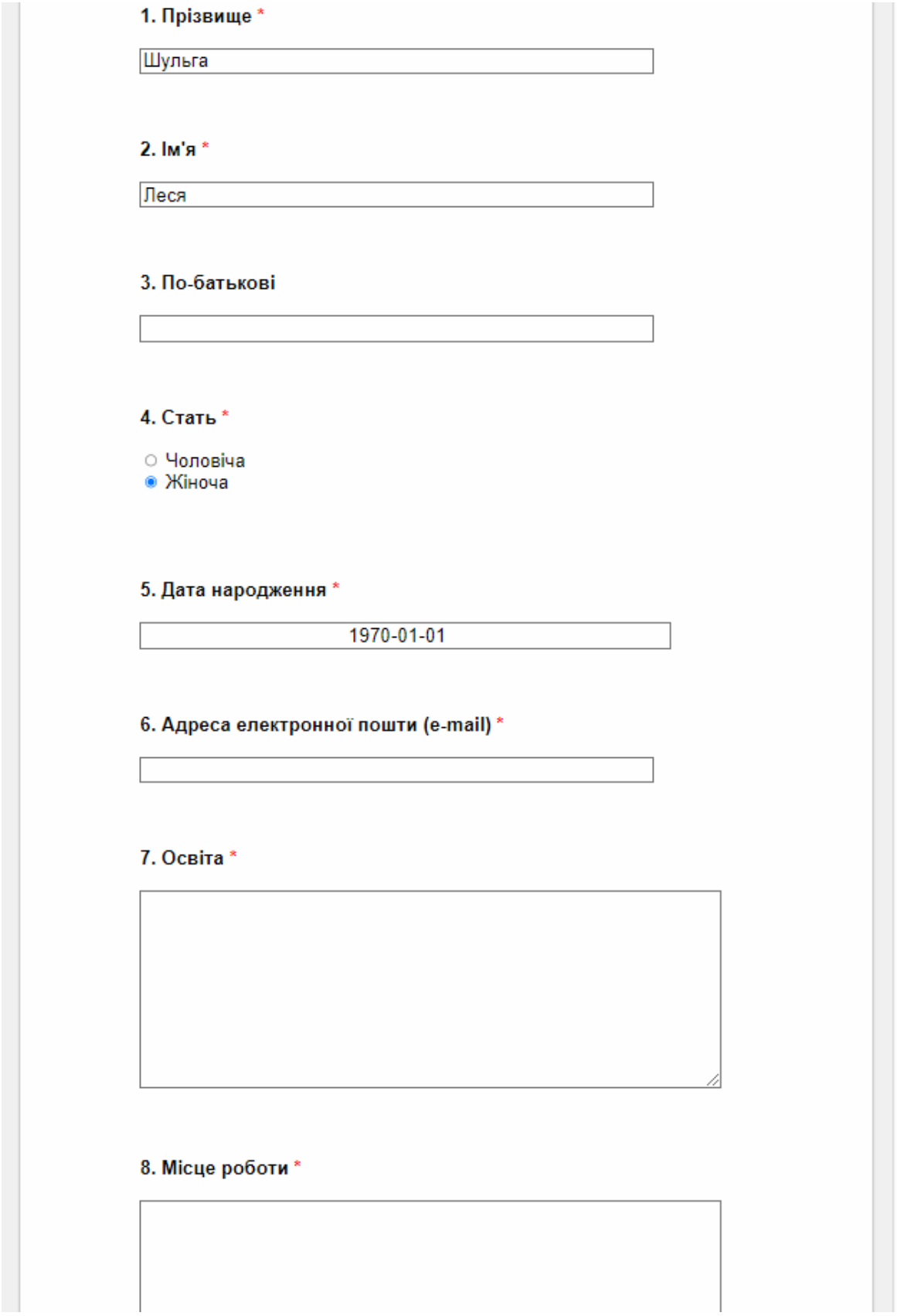


Рисунок Б.16 – Інтерфейс сторінки "Записатись на курс"

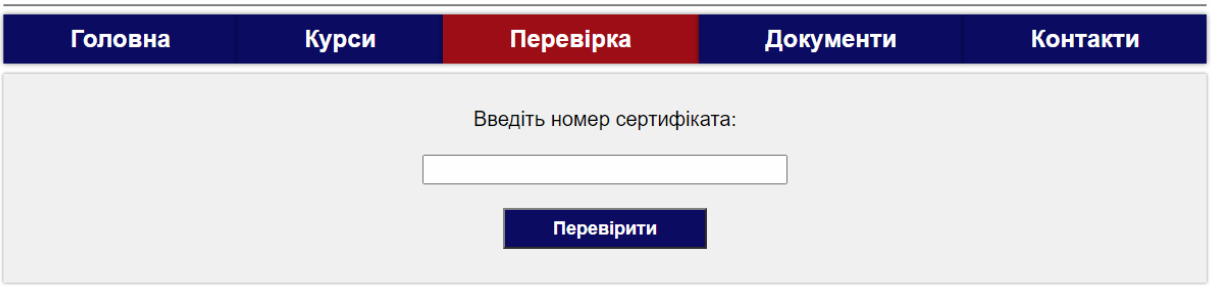


Рисунок Б.17 – Інтерфейс сторінки "Перевірка сертифікату" (введення номера сертифікату)

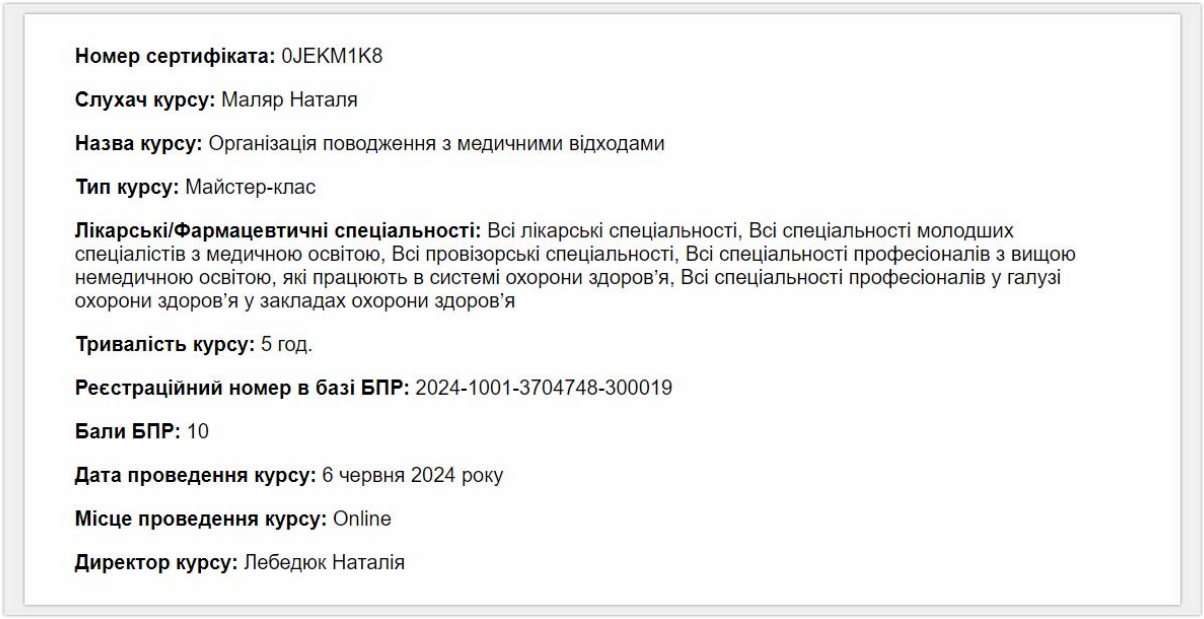


Рисунок Б.18 – Інтерфейс сторінки "Перевірка сертифікату" (перегляд детальної інформації про сертифікат)

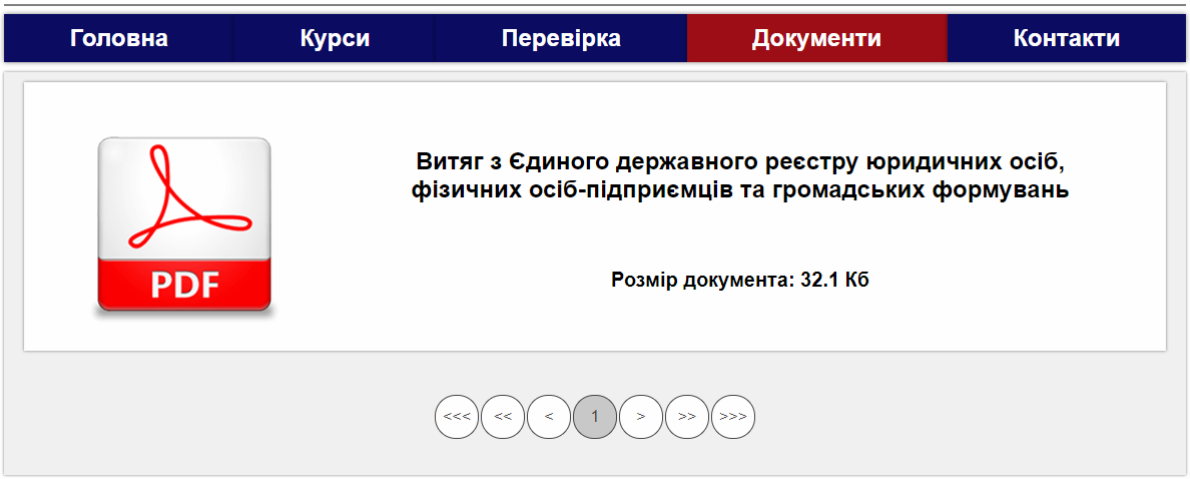


Рисунок Б.19 – Інтерфейс сторінки "Документи" (перегляд списку документів)

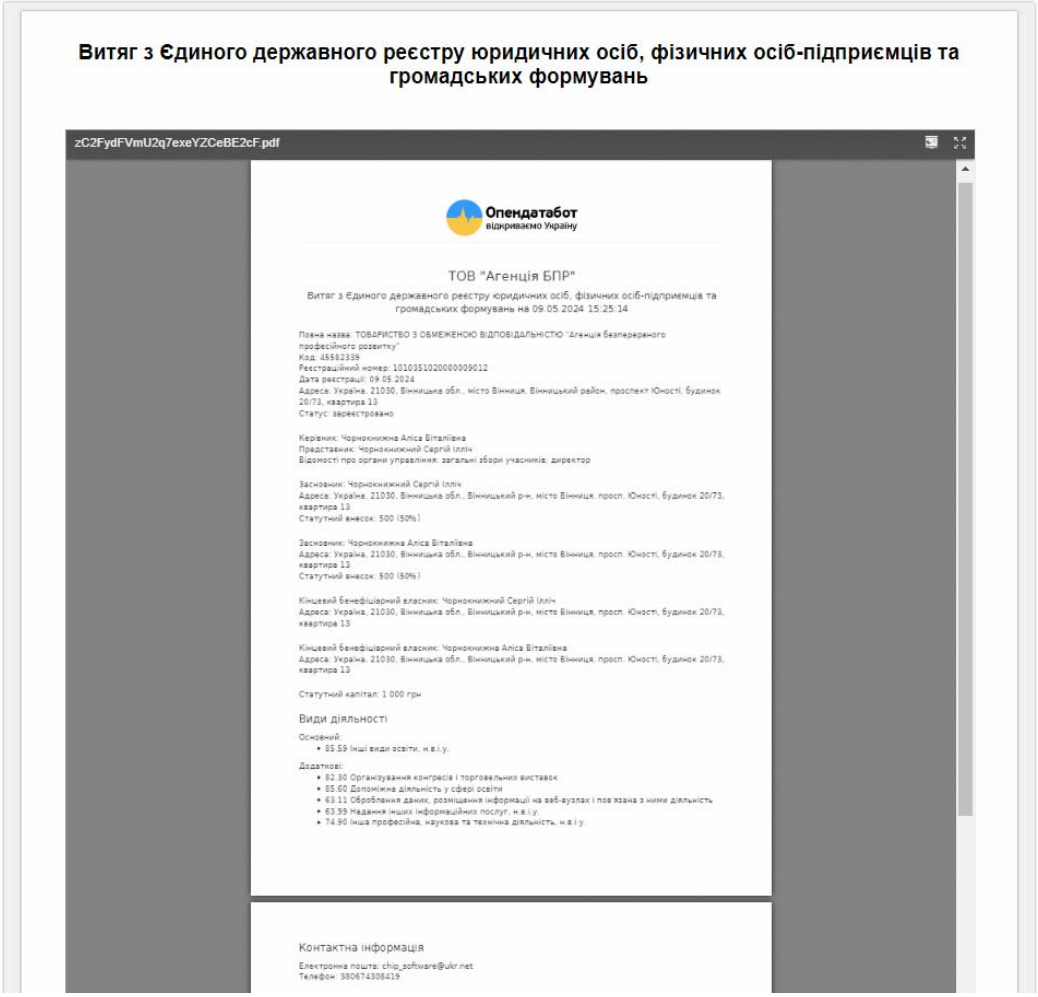


Рисунок Б.20 – Інтерфейс сторінки "Документи" (перегляд документу)

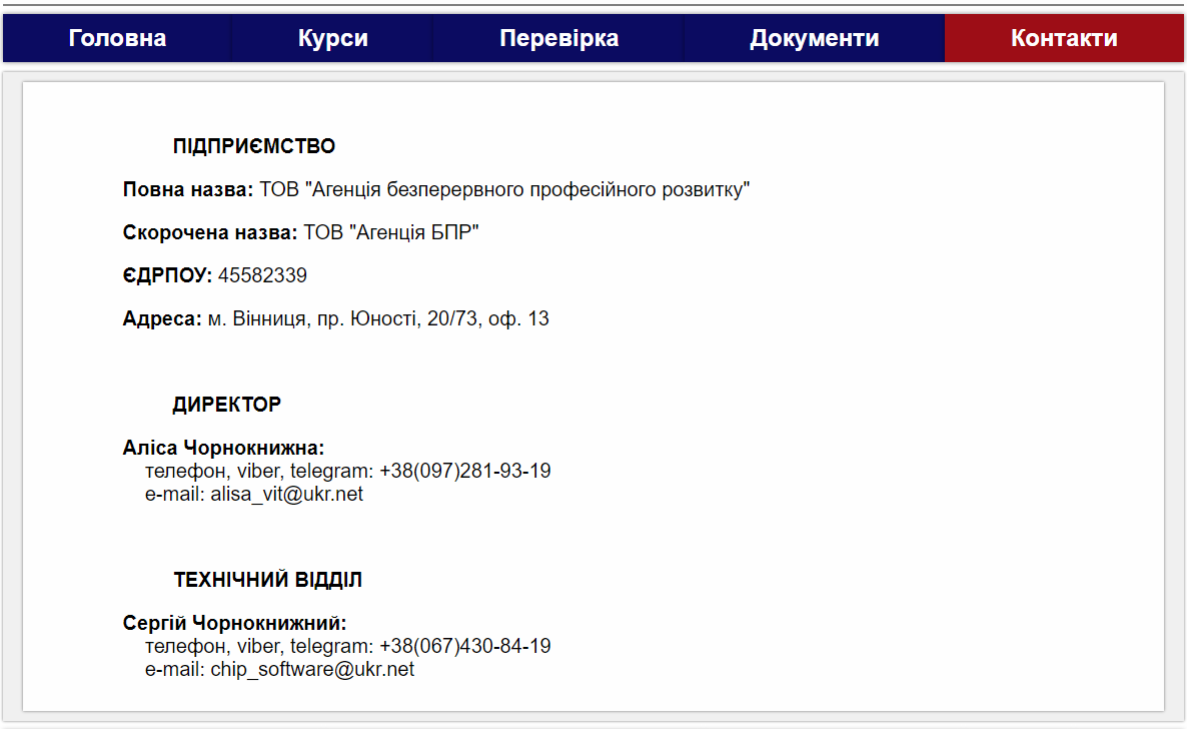


Рисунок Б.21 – Інтерфейс сторінки "Контакти"

Додаток В (обов'язковий)

Протокол перевірки навчальної (кваліфікаційної) роботи

ПРОТОКОЛ

ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Система автоматизації процесів провайдера безперервного професійного розвитку медичних працівників.

Тип роботи: магістерська кваліфікаційна робота

Підрозділ: кафедра автоматизації та інтелектуальних інформаційних технологій, факультет інтелектуальних інформаційних технологій та автоматизації

Показники звіту подібності Unicheck

Оригінальність 94,6% Схожість 5,4%

Аналіз звіту подібності (відмітити потрібне):

■ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на розгляд експертної комісії кафедри.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень

Особа, відповідальна за перевірку _____ Роман МАСЛІЙ
(підпис) (ім'я, ПРІЗВИЩЕ)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Сергій ЧОРНОКНИЖНИЙ
(підпис) (ім'я, ПРІЗВИЩЕ)

Керівник роботи _____ Євген ПАЛАМАРЧУК
(підпис) (ім'я, ПРІЗВИЩЕ)

Додаток Г (довідниковий)

Акт впровадження

ТОВАРИСТВО З ОБМЕЖЕНОЮ ВІДПОВІДАЛЬНІСТЮ «АГЕНЦІЯ БЕЗПЕРЕРВНОГО ПРОФЕСІЙНОГО РОЗВИТКУ»

Код ЄДРПОУ: 45582339
пр. Юності 20/73, оф.13, м. Вінниця, 21030
тел: (097) 281-93-19
e-mail: office@abpr.org.ua
web-site: https://abpr.org.ua
p/p: UA333052990000026003006107598
АТ КБ "ПРИВАТБАНК", МФО 305299

АКТ

№ 1 від 04.06.2024

Впровадження програмного продукту для автоматизації процесів провайдера БПР, розробленого в магістерській роботі студентом ВНТУ Чорнокнижним С.І.

Даний акт складений про те, що створений в магістерській роботі Чорнокнижного Сергія Ілліча програмний продукт, впроваджений у ТОВ «Агенція БПР» для автоматизації процесів провайдера безперервного професійного розвитку.

Впровадження прикладного програмного продукту у ТОВ «Агенція БПР» дає можливість автоматизувати діяльність провайдера безперервного професійного розвитку ТОВ «Агенція БПР».

Переданий автором пакет прикладної програми використовується на підприємстві співробітниками, які відповідають за реєстрацію заходів БПР в МОЗ та звітування за їх проведення.

Директор ТОВ «Агенція БПР» _____ Чорнокнижна А.В.