Хмельницький національний університет
Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

Програмний модуль запису на прийом до лікаря приватної клініки

Рівень вищої освіти Перший (бакалаврський)

Галузь знань 12 «Інформаційні технології»

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

Шифр КвРІПЗ. 180121.01.19.ПЗ

Виконав студент IV курсу, група ІПЗ-19-1 _____ А.О.Тропарчук
підпис Ініціали, прізвище

Керівник канд. техн. наук, доцент _____ Ю.В Форкун.
підпис Ініціали, прізвище

Нормоконтролер канд. техн. наук, доцент _____ І.В.Гурман
Підпис Ініціали, прізвище

До захисту допускаю:
Завідувач кафедри інженерії
програмного забезпечення _____ Л. П. Бедратюк
Підпис Ініціали, прізвище

8 червня 2023 р.

Хмельницький 2023

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій
Кафедра Інженерії програмного забезпечення
Рівень вищої освіти Перший (бакалаврський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Л. П. Бедратюк
06 03 2023 р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ)

Тропарчуку Андрію Олеговичу
Прізвище, ім'я, по батькові студента

1. Тема проєкту (роботи) Програмний модуль запису на прийом до лікаря приватної клініки

Керівник проєкту (роботи) Форкун Юрій Вікторович, канд. техн. наук, доцент
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання
Затверджена наказом ректора університету від 01.03.2023 р. № 5

2. Строк подання студентом проєкту (роботи) на кафедру 01.06.2023 р.

3. Вихідні дані до проєкту (роботи) Матеріали переддипломної практики, Методичні вказівки до виконання кваліфікаційної роботи

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Дослідження предметної області та постановка задачі, проектування програмного забезпечення, програмна реалізація системи, тестування програмної системи

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)
Презентаційні матеріали (слайди)

6. Консультанти розділів дипломного проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Гурман І.В., доцент кафедри ІПЗ	6.06.2023	7.06.2023
Антиплагіат	Гурман І.В., доцент кафедри ІПЗ	7.06.2023	7.06.2023

7. Дата видачі завдання «05» лютого 2023 р.

КАЛЕНДАРНИЙ ПЛАН

Назва етапів (розділів) дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1 Ознайомлення з тематикою дипломного проектування (ДП), визначення та узгодження індивідуальних тем ДП	05.02 – 14.02.2023	
2 Дослідження предметної області, в якій планується використання програмного засобу (ПЗ), визначення задач та вимог, розробка технічного завдання	15.02 – 28.02.2023	
3 Проектування програмного забезпечення	01.03 – 30.03.2023	
4 Програмна реалізація	01.04 – 20.04.2023	
5 Тестування програмного забезпечення	21.04 – 30.04.2023	
6 Написання вступу, загальних висновків, оформлення джерел посилання та додатків. Оформлення пояснювальної записки ДП згідно вимог стандартів	01.05 – 25.05.2023	
7 Попередній захист ДП	Травень 2023 (згідно графіка)	
8 Перевірка ДП на плагіат, нормконтроль, отримання відгуків та рецензій. Брошурування (зшиття) пояснювальної записки)	26.05 – 30.05.2023	
9 Підготовка до захисту та захист ДП	з 01.06.2023	

Студент _____ Пропарчук А.О
Підпис Ініціали, прізвище

Керівник проєкту (роботи) _____ Гурман І.В.
Підпис Ініціали, прізвище

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Програмний модуль запису на прийом до лікаря приватної клініки»

Автор проекту: Тропарчук Андрій Олегович

Керівник проекту: Форкун Юрій Вікторович

Пояснювальна записка: 80 с., 13 рис., 3 табл., 7 дод., 52 джерела

Графічна частина: 14 презентаційних слайдів

Ключові слова: AF, BR, CSS, CT, DBMC, HTML, JS, MVC, MYSQL, QA, QC, SQL, SATC, TC,TCS, TCPT,TP, TD

Мета кваліфікаційної роботи: розробка архітектури, компонентів програмного модуля та програмного застосунку запису на прийом до лікаря, яка б дозволяла здійснювати запис на прийом до лікаря.

Практичне значення полягає у автоматизації роботи реєстратури приватної клініки, завдання якої полягає у наповненні та обробці в інформаційній системі даних пацієнтів та лікарів. Система також максимально дозволяє спростити виконання основних операцій над такими об'єктами, а саме: редагування, видалення і створення нових записів.

Основна перевага даної системи полягає у гнучкості її налаштування, що дозволяє ефективно та максимально якісно і просто працювати з нею.

05.06.2023
Дата

Підпис

ВІДОМІСТЬ ДОКУМЕНТІВ

№ рядка	Формат	Позначення документа	Найменування документа	К-сть аркушів	№ екз.	Примітка
			Текстові документи			
1	А4	КвРІПЗ.180121.01.11.ПЗ	Пояснювальна записка	58		
2	А4		Завдання на кваліфікаційну	1		
			роботу			
3	А4		Анотація	1		
	А3		Графічні документи	3		
4	А4		Презентаційні матеріали	14		

Зм.	Арк.	№ докум.	Підпис	Дата	КвРІПЗ. 180121.01.19.ВД		
Виконав		Троварчук А.О.		3.06	Програмний модуль запису на прийом до лікаря приватної клініки **Відомість документів**	Літ.	Арк. 1 / Аркушів 88
Керівник		Форкун Ю.В.		3.06			
Рецензент						ХНУ, ІПЗ-19-1	
Н. Контр.		Гурман І.В.					
Зав. каф.		Бедратюк Л.П.		3.06			

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БД -	– база даних;
ІЛМ	– інформаційно-логічна модель;
ПЗ	–програмний застосунок;
СКБД	– система керування базами даних;
AF	– Abstract fabric (Шаблон Абстрактна Фабрика)
DBMS	– система керування базами даних;
BR	– Bug Report;
CSS	–мова оформлення стилю веб-сторінок;
CT	– Cloud Technology (Хмарна технологія) ;
HTML	– мова розмітки веб-сторінок;
MVC	–Model-View-Controler (архітектурний шаблон);
MYSQL	– система керування та сервер баз даних MySQL;
JS	– скриптова мова програмування;
QA	– Quality Assurance;
QC	– Quality Control;
SQL	– мова запитів до баз даних;
SATC –	– Software Assurance Technology Center;
TC	– Test Case;
TCS	– Test Case Specification;
TCPT	– Test Case Pass Time;
TP	– Test Plan;
TD	– Test Design.

ВСТУП

В нових умовах сьогодення майже у всіх сферах життя людини допомагає комп'ютер та інформаційні технології. Сучасний персональний комп'ютер є досить зручним, необхідним та унікальним засобом для обробки деякої інформації. Так у виробництві сучасний комп'ютер використовується для розробки деталей, їх проектування, конструювання певного виробу, та подальше його виробництва і його реалізацією. Музикантам він допомагає в написані музики, художникам створювати свої нові картини.

Зараз в кабінеті кожного окремого фахівця є свій персональний комп'ютер, який йому допомагає працювати людям дистанційно на відстані та дає можливість, навіть не виходячи з свого дому користуватися бібліотеками у різних країнах світу, використовувати різні технології, навчатись різним наукам, мовам, майстерностям тощо. З допомогою комп'ютера можна здійснювати оплату рахунків, оплату комунальних послуг, обирати та купувати товари в різних інтернет-магазинах, купувати та замовляти авіаційні, залізничні, та автобусні квитки.

Комп'ютер виконує певні операції та дії за допомогою спеціалізованого встановленого програмного забезпечення (ПЗ). Це може бути програмне забезпечення для перегляду та редагування текстів, зображень, їх друку, відтворення відео та медіа-файлів та багато іншого. Зараз такі функції можна використовувати не тільки за допомогою спеціалізованого програмного забезпечення, а й за допомогою різних веб-ресурсів. Веб-ресурс є зручнішими та доступнішими в користуванні, оскільки для їх перегляду та використання необхідно мати встановлений браузер на комп'ютері.

Відповідно, таким програмним забезпеченням може стати програмне забезпечення, що може автоматизувати роботу реєстратури в поліклініці, лікарні чи приватній клініці. Але, оскільки зараз існує багато різного роду операційних систем, що постійно оновлюють свої версії, то такий додаток було вирішено створити без привязки до певного типу операційних систем. Це робиться для того,

щоб наш програмний модуль мав змогу працювати на різних типах операційних системах, а користувач мав змогу доступу до нього з будь-яких пристроїв та операційних систем, без встановлення необхідного ПЗ, використовуючи тільки будь-який веб-браузер для перегляду веб-сторінок.

Метою створення такого веб-орієнтовного програмного модуля та розробка його інтерфейсу є:

– розробка бази даних для застосунку;
– автоматизація робочого місця працівників приватної клініки;
– можливість користувачам користуватись наданою інформацію;
– покращення забезпечення безпеки збереженості даних;
– зменшення затрат часу користувачів на оформлення різних документів.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз та визначення видів, методів та змісту інформаційного забезпечення предметної області

Приватна клініка – це лікувальний профілактичний заклад, основне призначення якої надання пацієнтам медичної допомоги стаціонарно. Спеціалізована клініка - це лікарня де надається стаціонарна допомога за однією визначеною спеціалізацією, в іншому випадку з багатьма різними спеціальностями – багатопрофільною. Багатопрофільні та спеціалізовані клініки зазвичай у своєму складі мають свої амбулаторії та поліклініки .

У таких лікарських закладах, їхні працівники реєстратури відповідають за графік прийому до лікаря, здійснюють запис пацієнтів, ведуть картотеку та ведуть спеціалізовану документацію.

Автоматизація роботі в цій сфері є досить доречною для здійснення таких дій, коли пацієнт навіть не виходячи з дому міг зробити запис на прийом до певного лікаря, дізнатися про графік роботи лікар та підібрати свій час відвідування найкращим чином.

Таким чином, у нашому проекті має бути передбачено режими адміністратора та користувача. Адміністратор має заносити інформацію про працівників поліклініки, лікарів, пацієнтів, а користувачі (реєстратори) мають здійснювати запис пацієнтів на прийом та створювати графік роботи лікарів.

База даних реєстратури потребує постійного оновлення, оскільки кожного дня йдуть прийоми замовлень, ведеться облік великої кількості пацієнтів та клієнтів. Тому найкраще створювати систему роботи у реальному часі, з постійним доступом до мережі інтернет. Таким чином проект має реалізуватися на основі хмарної технології. Хмарна технологія (Cloud Technology(CT)) – це загалом парадигма, яка передбачає віддалену обробку та збереження даних. Дана технологія надає можливість користувачам Інтернету доступ до ресурсів сервера

та його програмного забезпечення як деякого онлайн-сервісу. Тобто, якщо є підключення до мережі Інтернет то можна виконувати різні обчислення, опрацьовувати надані дані з використанням потужностей віддаленого сервера.

1.2 Аналіз досліджень, засобів розробки та існуючих рішень

Для збереження даних про графік роботи лікарів, та відомості про пацієнтів, інформацію про лікарські заклади, дані лікарів, а також швидкий і зручний доступ для перегляду такої інформації є доцільним використання бази даних. Таки чином, база даних складається з двох частин: збереження інформації та системи керування нею.

Для реалізації бази даних є велика кількість систем керування базами даних (СКБД), таких наприклад як MS Access, Oracle, DataBase, MS SqlServer, MySQL, PostrgresSQL.

СКБД MySQL має свої переваги над перерахованими вище:

- простота у розгортанні та використанні;
- MySQL підтримується необмежена кількість користувачів, які можуть одночасно працювати із базою даних;
- кількість збережених рядків у таблицях може досягати мільярди записів;
- висока швидкість виконання SQL команд;
- наявність простої і досить ефективної системи безпеки збереження даних та їх опрацювання.

База даних повинна приведена до 3 НФ за основними правилами нормалізації, що наведені нижче.

Перша нормальна форма (1НФ):

- кожна таблиця повинна містити основний ключ (мінімальний набір атрибутів), які однозначно ідентифікують запис;

– уникатись повторень певних груп (категорії даних, які зустрічаються певну кількість раз у різних записах) та вірно визначати неключові атрибути;

– вслативість атомарності - кожен атрибут має мати лише одне значення, а не деяку множину значень.

Друга нормальна форма (2НФ):

– таблиці бази даних повинні відповідати вимогам 1НФ;

– дані, які з'являються в декількох рядках повторно потрібно винести в окремі сутності.

Третя нормальна форма (3НФ):

– таблиці бази даних повинні відповідати всім вимогам 2НФ;

– кожне поле, яке залежить від основного ключа чи від певного іншого поля також має виноситись в окрему таблицю БД.

Для розробки веб-сайтів існує велика кількість мов програмування таких як: HTML, Java, Python, PHP, Ruby тощо. Також при проектуванні використовуються різні фреймворки: Jumla, Django, Symphony, Drupal, Simple, Ruby та інші. Більшість наведених фрейморків мають досить великий функціонал та багато різни модулів, які навантажують сервер – це і є їх головний недолік. PHPixie – має невеликий набір можливостей, проте має чудовий захист та добре задокументований код, тому є доцільність в використанні такого фреймворку.

Розглядаючи різні аспекти використання мови PHP, можна виділити наступні основні її переваги над іншими мовами:

– традиційність мови ООП;

– простота мови;

– ефективність використання мови;

– безпека мови;

– гнучкість при проектуванні коду.

Традиційність мови. Синтаксис та конструкції мови PHP включають багато елементів, що зустрічаються у мовах програмування C, C++, Perl. PHP є мовою

програмування з універсальним синтаксисом з пристосуванням до веб-програмування.

Простота мови. Мова PHP та її скрипти можуть бути вбудовані безпосередньо в html-код веб-сторіно сайтук, що обробляються PHP-інтерпретатором. Сама мова програмування PHP містить величезну кількість різноманітних функцій, що саме позбавляє необхідності писати багаторядкові програмні скрипти для виконання деякого завдання. Головне завдання для програміста – правильно обрати функцію відповідно до певного конкретного завдання. Окрім того, не потрібно додатково завантажувати додаткові бібліотеки та вказувати спеціальні параметри при компіляції коду.

Розглядаючи мову програмування Python для веб-сторінок можна серед основних її переваг можна навести настуні:

- чистий синтаксис мови Python;
- Python має зрозумілу і просту синтаксичну структуру, що робить його легким для вивчення навіть для початківців;
- читабельність коду Python сприяє розумінню програм і спрощує підтримку інших розробників;
- льність: Python є загальнопризначеною мовою програмування, що дозволяє розробляти різноманітні типи програм, включаючи веб-програми, наукові обчислення, штучний інтелект, автоматизацію завдань і багато іншого;
- Велика кількість сторонніх бібліотек і модулів робить Python потужним інструментом для різних задач;
- Python має велику спільноту розробників, яка активно підтримує мову.
- Python підтримується на багатьох операційних системах, таких як Windows, macOS і різні дистрибутиви Linux. Це дозволяє розробникам писати один раз і виконувати програми на різних платформах без необхідності великих модифікацій.
- Python має велику кількість сторонніх бібліотек і модулів, що дозволяє розробникам ефективно виконувати різні завдання.

– переносимість та адаптованість програм;

– стандартний дистрибутив містить велику кількість потрібних модулів;

– можливість використання мови Python в діалоговому режимі;

– стандартний дистрибутив містить зручне, але поряд із тим досить потужне середовище для розробки, з назвою IDLE, яке теж написане на мові програмування Python;

– досить зручний для розв'язання різноманітних математичних проблем.

Прикладами подібних сайтів в нашій країні можна назвати веб-сайт medics.ua, веб-сайт https://helsi.me/ та веб-сайт poliklinika5.lviv.ua, головні сторінки їхніх сайтів наведені на рисунку 2.1 [9] та рисунку 2.2 [10] та 2.3 [11]:

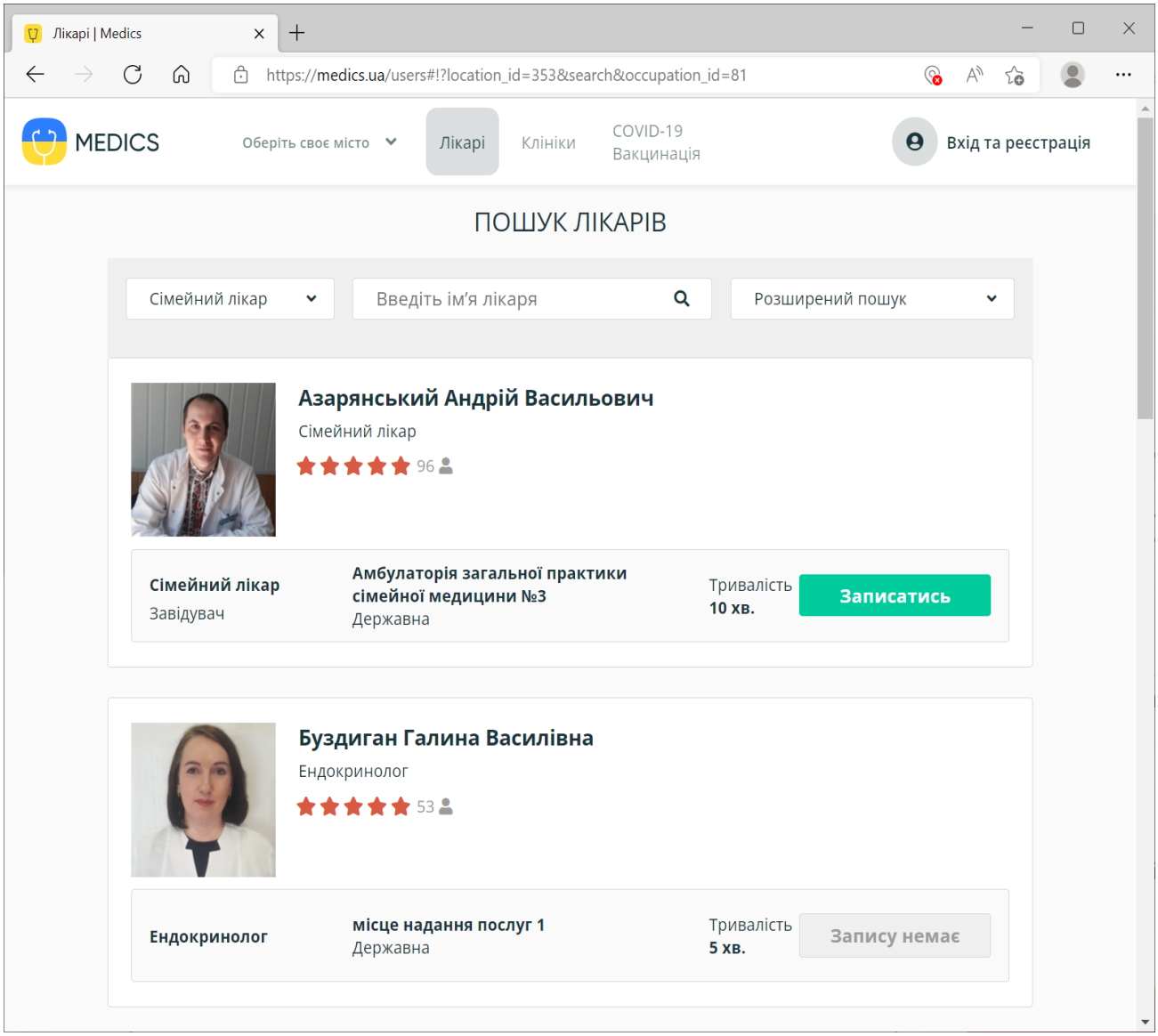


Рисунок 1.1 – Веб-сайт medics.ua

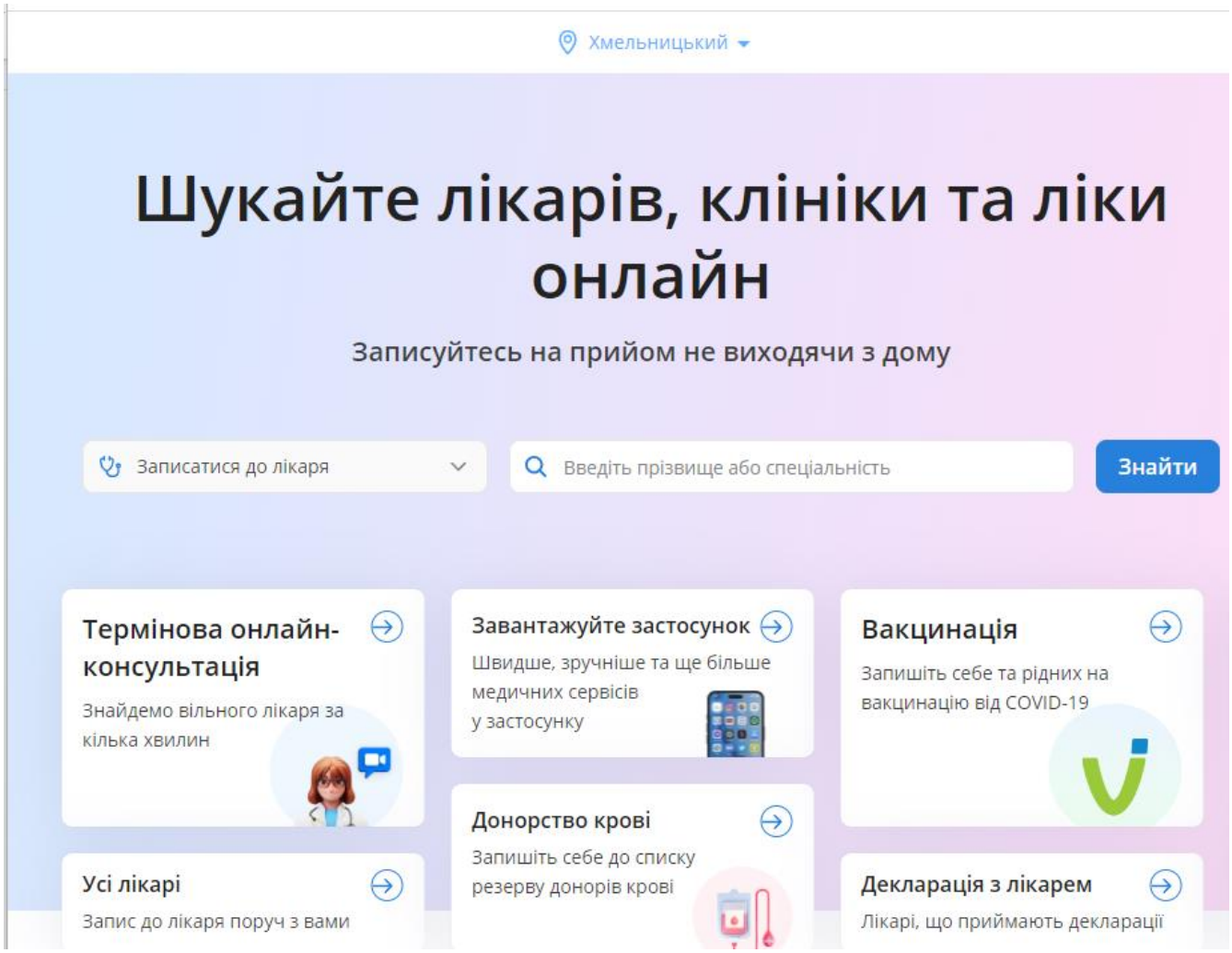


Рисунок 1.2 – Веб-сайт https://helsi.me/

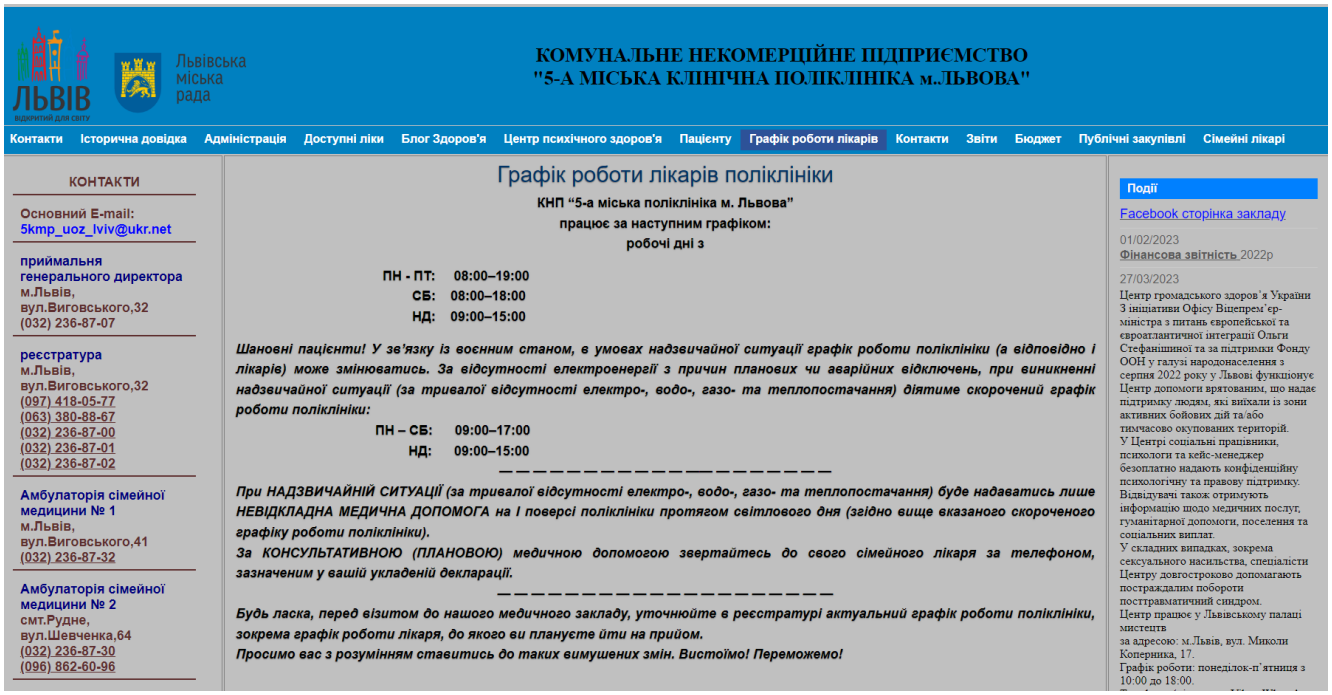


Рисунок 1.3 – Веб-сайт poliklinika5.lviv.ua

1.3 Постановка задачі та перелік задач для реалізації

Метою розробки даного проекту є створення веб-системи управління реєстратурою лікувального закладу.

У нашому застосунку мають бути передбачені режим адміністратора та режим користувача. Адміністратор має заносити детальну інформацію про пацієнтів, лікарів, клієнтів, а співробітники-реєстратори повинні вносити дані клієнтів про прийом до лікарів, заносити графік роботи лікаря тощо.

Завданням проекту є також створення простого інтерфейсу для дій введення та виведення даних їх редагування, збереження, обробки та аналізу у базі даних.

В проекті необхідно передбачити такі дії та ресурси:

- авторизацію користувача адміністратора та користувача в системі;
- інтуїтивно зрозумілий інтерфейс програмного модуля;
- сторінка для адміністрування веб-сайту;
- пошукову система, за наперед заданими фільтрами;
- зручний запис на прийом та їх редагування;
- перегляд та редагування даних в базі даних;
- формування різноманітних звітів в залежності від наявних отриманих результатів.

Також було розроблене технічне завдання, яке розміщене у додатку А і його можна використовувати як основу для нашої подальшої розробки веб-застосунку для управління реєстратурою лікувального закладу.

База даних сервісу здійснює постійне оновлення, адже кожного дня в клініці приймаються багато різних записів на прийом та амбулаторне обслідування, тому необхідно також вести облік значної кількості персоналу та пацієнтів.

Наш проект буде реалізований в вигляді веб-застосунку на мові програмування php та фрейморка WordPress, саме для того, щоб не обмежувати наш застосунок в певній операційній системі. База даних буде створена в СКБД MySQL, як найбільш прийнятною та поширеною. Для розробки даного проекту

нами була обрана клієнт-серверна архітектура, в якій у ролі клієнтської частини виступає веб-браузер, а серверною частиною веб-сервер, щоб міг бути доступ до сервісу в будь-який момент часу та без встановлення додаткового ПЗ.

1.4. Висновок першого розділу

У першому розділі проведено дослідження предметної області. Здійснено аналіз та визначення видів, методів та змісту інформаційного забезпечення предметної області. Проведено аналіз та визначення видів, методів та змісту інформаційного забезпечення предметної області. Здійснено постановку задачі та окреслено перелік задач для реалізації.

2 ПРОЕКТУВАННЯ ПРОГРАМНОГО МОДУЛЯ

2.1 Вибір типу архітектури та зразків проектування

Як було вказано попередньо, в даній системі буде використана архітектура «клієнт-сервер».

Дана технологія «клієнт-сервер» широко застосовується при роботі з різноманітними СКБД та базами даних в мережі. Ця технологія відома вже досить давно і найчастіше застосовувалась у великих компаніях. Сьогодні, з розвитком мережі Internet, дана технологія найчастіше приваблює погляди проектувальників та розробників програмного забезпечення, оскільки в світі відбувається нагромадження величезної кількісті інформації з різних питань яка зберігається в різних базах даних. Технологію клієнт-сервер загалом можна описати наступним алгоритмом:

- на стороні клієнта формується та відправляється новий запит до бази даних серверу БД, а точніше до програми, яка обробляє ці запити;
- сервер здійснює маніпуляції з даними базами даних, які розташовані на сервері, у відповідності до запиту та формує результат який передає його клієнту;
- далі клієнт отримує результат та відображає його на пристрої і очікує подальших дій від користувача. Цей цикл повторюється доті, доки користувач не завершить роботу з програмою.

Перевага архітектури клієнт-сервер полягає у наступному:

- відсутності дублювання програмного коду функціоналу сервера у клієнтських програмах;
- низький рівень вимог до технічного обладнання клієнтського обладнання, через те, що всі маніпуляції та дії відбуваються на сервері;
- всі дані, що зберігаються на самому сервері, який як правило, має набагато кращий захист ніж більшість клієнтських терміналів.

Для проектування структури роботи системи нами було обрано патерн проектування Model View Controller (MVC).

MVC – це такий шаблон проектування, який являє собою зразок певної архітектури програмного забезпечення, що заснований по принципу поділу представлення даних та функціоналу, в якому ці дані формуються та обробляються.

Мета цього шаблону — гнучка розробка програмного забезпечення, який здатен полегшувати подальші зміни програмного коду чи розширення функціоналу програм. Цей шаблон також може надавати можливість повторного використання його окремих компонентів. Крім того, використання цього патерну у великих системах дозволяє привести їх до певної впорядкованості його структури та робить код зрозумілішими завдяки структурованості.

Архітектурний шаблон Модель-Вид-Контролер (Model View Controller (MVC) робитть поділ складових програмного забезпечення на три різних частини. У цій трійці до обов'язків компоненту «Модель» (Model) належать частини зберігання даних та забезпечення інтерфейсу доступу до них. Компонент «Вигляд» (View) несе код, який відповідальний за надання таких даних користувачу. Компонент «Контролер» (Controller) здійснює ууправління компонентами, отримує різні сигнали через реакції на дії користувача та передає інформацію про зміни компоненту «Модель». Таке подання внутрішньої структури в цілому дозволяє розділити модуль на самостійні частини та розподілити відповідальність та контроль між різними її компонентами.

Модель MVC поділяє проектовае програмне забезпечення на три незалежні частини: компонент введення/виведення даних, компонент обробки даних та компонент взаємодії модулів. Модель, як було відмічено вище, встановлює ядро даних та основний функціонал обробки даних. Також компонент «Модель» є незалежним від процесу введення/виведення даних. Компонент виведення «Вигляд» може містити декілька взаємопов'язаних областей, такі як таблиці, поля форм де відображається інформація. У функції компонента «Контролер» входить відповідальність моніторингу за подіями, які виникають в результаті певних дій користувача (зокрема, зміна положення курсора мишки, натиснення певної

кнопки чи введення даних в поле на формі).

Зареєстровані події модуля передаються в різні запити, які спрямовуються компонентам «Моделі» чи об'єктам, які відповідають за відображення даних. Відокремлення «Моделі» від «Вигляду» дозволяє використовувати різні компоненти незалежно для відображення певної інформації. Отже, якщо користувач з допомогою компоненти «Контролер» внесе зміни до компоненти «Модель», то інформація, яка подана одним чи декількома візуальними компонентами, автоматично відкоригується відповідно до внесених змін.

Стандартне програмне забезпечення, яке реалізується за технологією клієнт-сервер, має добру масштабованість (через ефективне використання наявного апаратного та програмного забезпечення). Тож наявна стійкість у роботі, захист від різного роду несанкціонованого доступу та продуктивність при роботі з великими проектами з великими базами даних.

Конкретність роботи модуля залежить від того де знаходиться сервер і яким чином клієнт під’єднаний до цього сервера. Користувач на свому комп’ютері у браузері заповнює обрану форму та виправляє подальшу дію. Браузер при натисненні певної кнопки на формі передає дані із цієї заповненої форми, або знову відображає отримані в результаті певної операції. При цьому не важливо, як до мережі під’єднаний клієнт. Він може бути віддаленим користувачем і з’єднуватися будь-яким способом. Модуль приймає дані, здійснює їх перевірку та формує запит до СКБД та отримує від нього результуючі дані. На самому сервері зберігається база даних, яка змінюється за запитами клієнта. У цьому режимі роботи забезпечується досить високий рівень безпеки даних бази даних, як від збоїв обладнання, різних сторонніх програм, так і від несанкціонованого доступу користувачів. Забезпечується висока продуктивність роботи, навантаження на мережу невисокі, проте зростають вимоги до самого сервера.

При проектуванні використовується шаблон проектування “Abstract fabric”. Abstract fabric — це шаблон проектування, який забезпечує інкапсуляцію окремих модулів під єдиною схемою, пропускаючи при цьому їхню деталізацію. Шаблон

належить до класу твірних шаблонів.

У таких типових випадках застосування, клієнтський код створює певну реалізацію абстрактної фабрики, а далі використовує сам загальний універсальний інтерфейс такої фабрики, для створення усіх екземплярів та об'єктів, які є частиною схеми. Тут клієнтський код не знає (не приймає до уваги), які саме об'єкти він отримує від таких фабрик, оскільки використовується універсальний інтерфейс при їх створенні. Цей шаблон розмежовує деталі реалізації об'єктів від їх конкретного використання в самому коді, тому що при створення об'єкта це здійснюється за допомогою різних методів, які забезпечуються самим інтерфейсом фабрики.

Розширення, в свою чергу, повинні певним чином реалізувати деякий загальний інтерфейс та можуть містити, по суті, різні інструменти. Однак, практично, кожне розширення працює з даними, де введено додатковий рівень абстракції, який описує даний механізм. Завдяки цьому, всі розширення працюють з даними уніфіковано та в єдиному контексті, що є досить важливим елементом. Детальний опис цих рішень подано у розділі детального проектування.

У більшості випадків розширення містять у собі контролери та подання, тому на першому кроці потрібно заставити працювати контролер, а також та динамічно завантажені збірки (зокрема зі збірки, на яку немає явних посилань в проекті project.json, основного веб-додатку та, зі збірки яка завантажується з каталогу після старту з наявними розширеннями.

Для реалізації інтерфейсу IAssemblyProvider достатньо скопіювати збірки з DefaultAssemblyProvider та додати до них ті наявні збірки, що завантажені динамічно з каталогу з необхідними розширеннями (ModularApp.ExtensionAssemblyProvider). Далі необхідно зареєструвати нову реалізацію в настройках сервісу ConfigureServices (ModularApp.Startup). Єдиний момент, що необхідно врахувати, це те, що за замовчуванням MVC буде шукати контролери в збірках, які посилаються на інші системні класи секції Model при даному підході.

Дизайн нашого сайту потрібен досить простий та лаконічний. Зокрема,

оформлення його в світлих тонах , де присутні світлі кольори, що не вкидаються в вічі, не заважають роботі з самим сайтом та не відволікають самого користувача від запропонованих йому товарів на сайті. Головна задача, це - це простота в навігації сайту для користувача, оскільки він може швидко втратити інтерес до самого сайту та інформації розміщеною на ньому, також якщо буде нерозбірливо написана надана інформація. Якщо все скидане один на одне, а колір шрифта також майже співпадає з фоном сторінки то це буде відштовхувати користувачів від сайту, а не навпаки втримувати чи заволікати. Тому дуже важливою є задача щоб вся інформація, яка розміщена на сайті була відображена сприйнятним та розбірливим шрифтом, як для зручності так і більш продуктивної роботи, що в наслідку приведе для покращення роботи поліклініки. Адже, якщо інтерфейс такого сайту не приваблює користувача, а відштовхує його, то користувач навряд чи щось замовить на ньому.

Існує декілька видів та підходів до побудови сайту. Вони називаються логічними структурами сайту, оскільки назва їхньої структури відповідає їх застосуванням та використанням. Логічні структури сайтів поділяють на лінійну структуру, лінійну структуру з різними альтернативами та варіантами, лінійну структуру з розгалуженнями, деревовидну структуру, а також решітчасту структуру сайту.

При створенні нашого інтернет-застосунку ми використаємо деревовидну структуру для побудови сайту, оскільки вона нам найбільше підходить для реалізації наших завдань, а саме реалізацію простоти, навігації та швидкість користування сайтом.

Деревовидна структура на сьогодні досить популярна, оскільки вона є досить зручною та зрозумілою в користуванні. Деревовидна структура, по суті означає, що при вже вході на головну сторінку сайту користувач опиняється перед дилемою: «куди піти далі та який розділ необхідно обрати». Така логічна структура підходить для реалізації майже всіх типів такого роду сайтів, оскільки одразу чітко видно всі розділи сайту, а тому користувач не втрачаючи часу на пошуки може

відкрити та продивитись необхідну йому інформацію. Після переходу у потрібний йому розділ користувач підбирає необхідний йому підрозділ і, далі по ієрархії поки не досягне необхідного йому рівня. Реалізація такої моделі користування веб-сайтом є дуже швидкою та зручною, оскільки користувач майже одразу потрапляє на потрібну йому сторінку.

Відкривши наш інтернет-застосунок користувач відразу бачить головну сторінку, на якій відображені усі головні елементи: блок реєстрація користувача, новини та останні події, пропоноване меню, де відображаються наявні лікарі, амбулаторії, розділи та підрозділи та пошук по інтернет-сату. Далі користувач може обирати необхідні йому розділи, або пропоноване меню.

2.2 Опис декомпозиції

Декомпозиція – це такий метод розкриття структури системи, при якому за певною ознакою її ділять на окремі складові.

Декомпозиція використовується при побудові «дерева цілей» та для прив’язки генеральної мети із способами та методами їх досягнення, які сформульовані у вигляді завдань окремими виконавцями.

Розглянемо загальні технологічні аспекти побудови "дерева цілей". Не існує таких універсальних методів побудови «дерева цілей». Його способи побудови залежать від певного характеру цілі, який обраний методологічний підхід, а також залежить від того, хто розробляє це "дерево цілей", та як він представляє собі поставлені йому завдання, і як він бачить їхній взаємозв'язок між собою.

Поставлену нам задачу було розділено на декілька модулів:

- модуль навігації, який включає в себе інтуїтивно зрозумілі для користувача головне меню, панелі інструментів, різні кнопки та контекстні меню для кожного з блоків модуля;
- модуль редагування даних, який містить усі потрібні елементи інтерфейсу, за допомогою яких дозволяється додавати, змінювати, редагувати та

видаляти дані, призначений для зміни інформації у базі даних;

- модуль фільтрації який дозволяє шукати інформацію відповідно до запитів користувачів і призначений для коректного відображення даних;
- модуль перегляду який включає в себе таблиці та поля з інформацією та графік роботи лікаря;

Модуль роботи адміністратора реєстратури забезпечує можливість виконання робіт створення, перегляду, обробки, редагування та видалення інформації про працівників, пацієнтів, прийомів та графіку роботи лікаря. Виходячи з поставленої мети декомпозиції, декомпозуємо модуль до таких функцій: "Створення інформації про прийом пацієнта", "Перегляд прийому лікарів", "Створення нового юзера", "Корегування даних", "Корегування інформації про прийом". Ці функції отримують введені адміністратором дані та передаються в БД. Модуль роботи з базою даних забезпечує можливість опрацювання та вибірки,та для роботи з редагування, перегляду І видалення даних адміністратором системи. В загальному база даних призначена для зберігання даних про працівників, лікарів, пацієнтів, тарифних планів та даних про них. На етапі програмування вибірка здійснюється за допомогою мови SQL-запитів.

Для запобігання проблем у зв’язку з внесенням некоректних даних, особливо при роботі з базою даних модуля, зв’язками між таблицями, ключами , необхідно створити певну систему обмежень та передбачити роботу загальної логіки програми.

У програмному модулі необхідно провести декомпозицію таким чином, щоб окремі елементі працювали як можливо максимально незалежно один від одного. В існуючих методів проведення декомпозиції нами було обрано модель потоку даних, в основі якої лежить розбиття системи на функції. Результатом нашої декомпозиції є модель системи, яка складається з ієрархічно впорядкованих наборів діаграм, що спрощуються із кожним рівнем ієрархії.

На початку створимо контекстну діаграму, яка описує даний програмний модуль в цілому та зв'язок з предметною областю. Від зовнішніх сутностей БД до

програмного модуля поступають такі сигнали та дані:

– дані про записи на прийом до лікарів клініки;
– дані про лікарів клініки та їх графік роботи;
– дані про клієнтів клініки;
– інформація про спеціалізацію лікарів клініки;
– графіки роботи кожного лікаря клініки;
– дані про кабінети клініки.

Система повертає наступні інформаційні дані:

– інформація про часи роботи клініки і її підрозділів;
– дані з інформацією про лікарів;
– дані з інформацією про послуги клініки;
– дані про часи роботи амбулаторій;
– інформацію про часи прийому аналізів;
– доступні часи для прийому лікарем пацієнта;
– дані з інформацією про чачи прийому адміністрації;
– відгуки пацієнтів про лікарів клініки;
– рейтинги для кожного лікаря.

2.3 Опис залежностей

Усі залежності зобразимо у вигляді діаграми UML-діаграми (Component diagram). Діаграма компонент загалом відображає залежності наявні між компонентами програмного забезпечення з включенням компонентів вихідного коду, різні бінарні компоненти, а також про інші компоненти які виконуються програмному модулі.

Сюди входять такі представники: адміністратор, працівник, пацієнт та направлення. Діаграма компонентів наведена на рисунку 2.1:

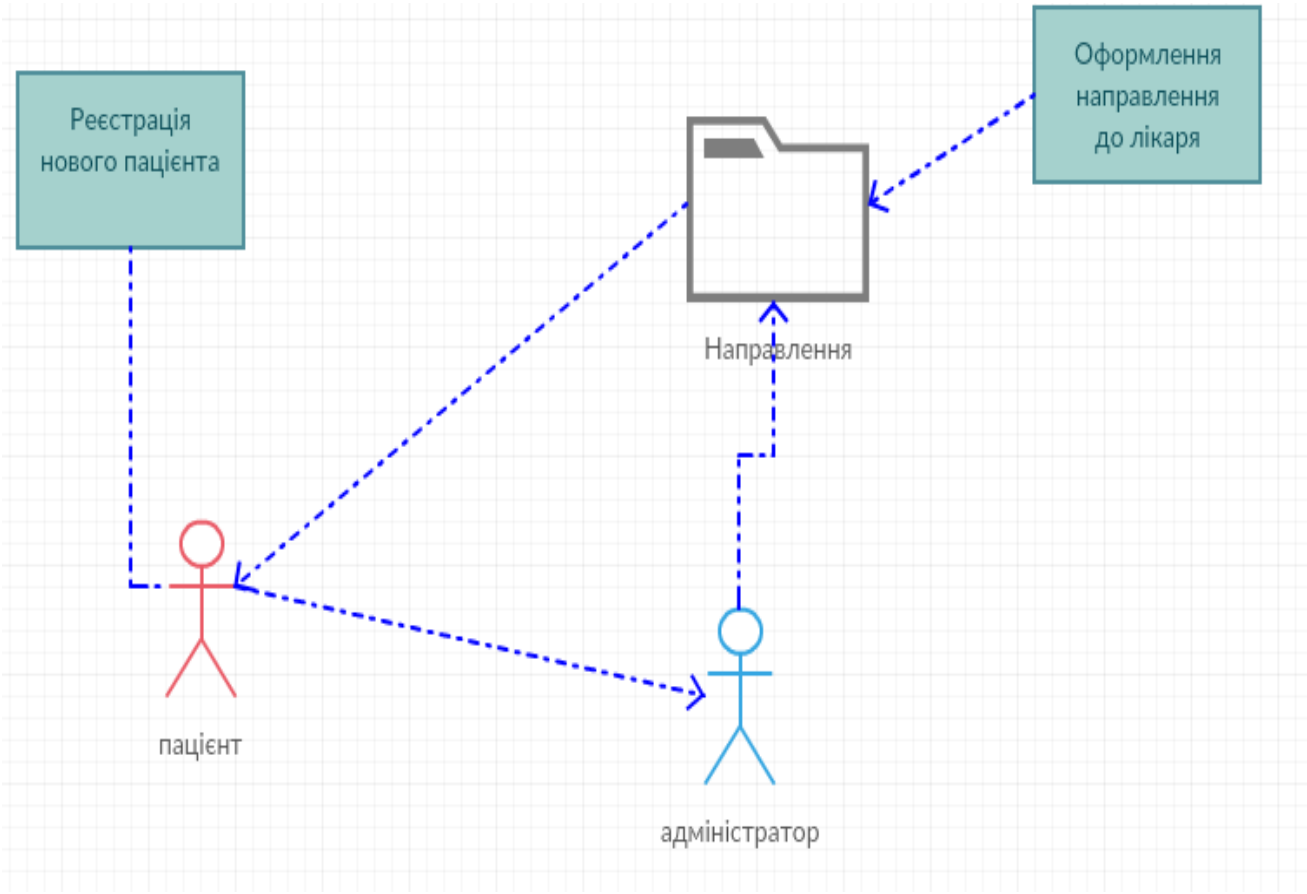


Рисунок 2.1 – Залежність між компонентами діаграми

Основний модуль компонента «Клієнт» керує такими модулями: інтерфейсів, звітів та модулем візуалізації (Рисунок 2.2):

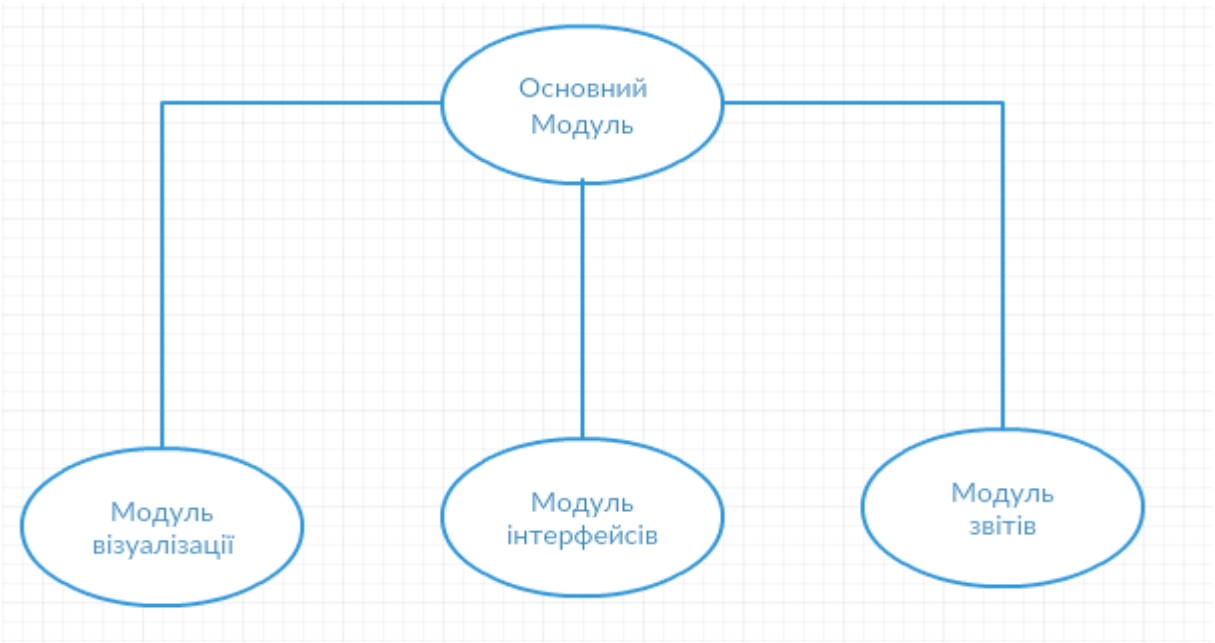


Рисунок 2.2 - Залежності між модулями пакету «Пацієнт»

Файл-серверна архітектура висуває певні вимоги до користувацької частини, хоч зі сторони користувача необхідна наявність лише довільного браузера. Так архітектура проекту вимагає також коректного відображення усіх стилів фреймворку та виконання JQuery-скриптів. найоптимальннішим є використання таких версій браузерів як Chrome, Opera, Edge, Dolphin тощо.

Для успішного функціонування браузера з відкритим кодом характеристики комп'ютера повинні відповідати таким вимогам:

- процесор з тактовою частотою не нижче 1,3 ГГц;
- оперативна пам'ять – 1 Gb та вище;
- об'єм вільного місця на диску не менше 150 Mb;
- операційна система з наявним графічним інтерфейсом;
- маніпулятор мишка (тачпад), клавіатура;
- наявність підключення до мережі Інтернет.

Для початку роботи модуля програми роботи користувач необхідно встановити будь-який веб-браузер. Далі необхідно ввести у поле пошуку адресу URL посилання на веб-сайт та натиснути кнопку «Перейти».

2.4 Опис інтерфейсів

Опис інтерфейсів зобразимо у нашій моделі у вигляді «с». До цих інтерфейсів буде належати вибір певного лікаря, вибір певного відділення, опрацювання даних лікаря, опрацювання даних пацієнтів, а також додаткові послуги.

Інтерфейсу у нас буде підпорядкована сутність, а цій сутності – модулі, контролери, генератори, сервіси та також граничні сутності. Крім того,у пректі наявна база даних. Шаблоном проектування нами обрано патерн «Abstract Fabric».

Розроблена діаграма інтерфейсів програмного модуля прийому лікарів наведена на рисунку 2.3:

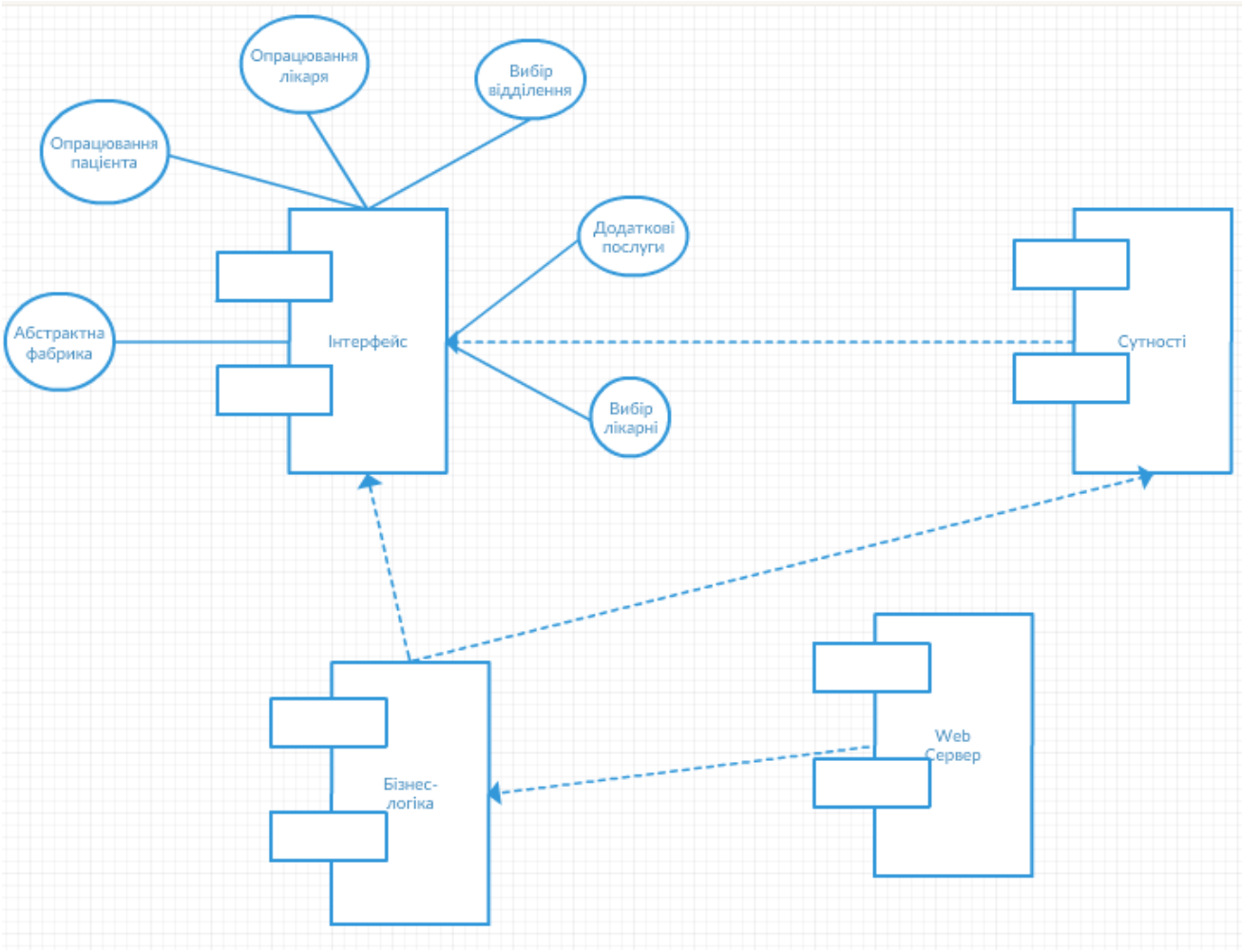


Рисунок 2.3 – Опис інтерфейсів програмного модуля

2.5 Детальне проектування та розробка модулів застосунку модулів

При реалізації детального проектування застосунку необхідно збудувати модульну модель. Під модульною моделлю розуміється реальна модель проектованої прикладної системи. Процес побудови включає в себе уточнення моделі бази даних з обов'язковою нормалізацією для подальшої генерації SQL-запитів; уточнення структури користувальницького інтерфейсу; побудова структурних схем, що відображають логіку роботи призначеного для користувача інтерфейсу та модель бізнес-логіки. Загальний алгоритм роботи модуля «Оформлення направлення» програмного застосунку представлений у вигляді «діаграми послідовності» на рисунку 2.4.

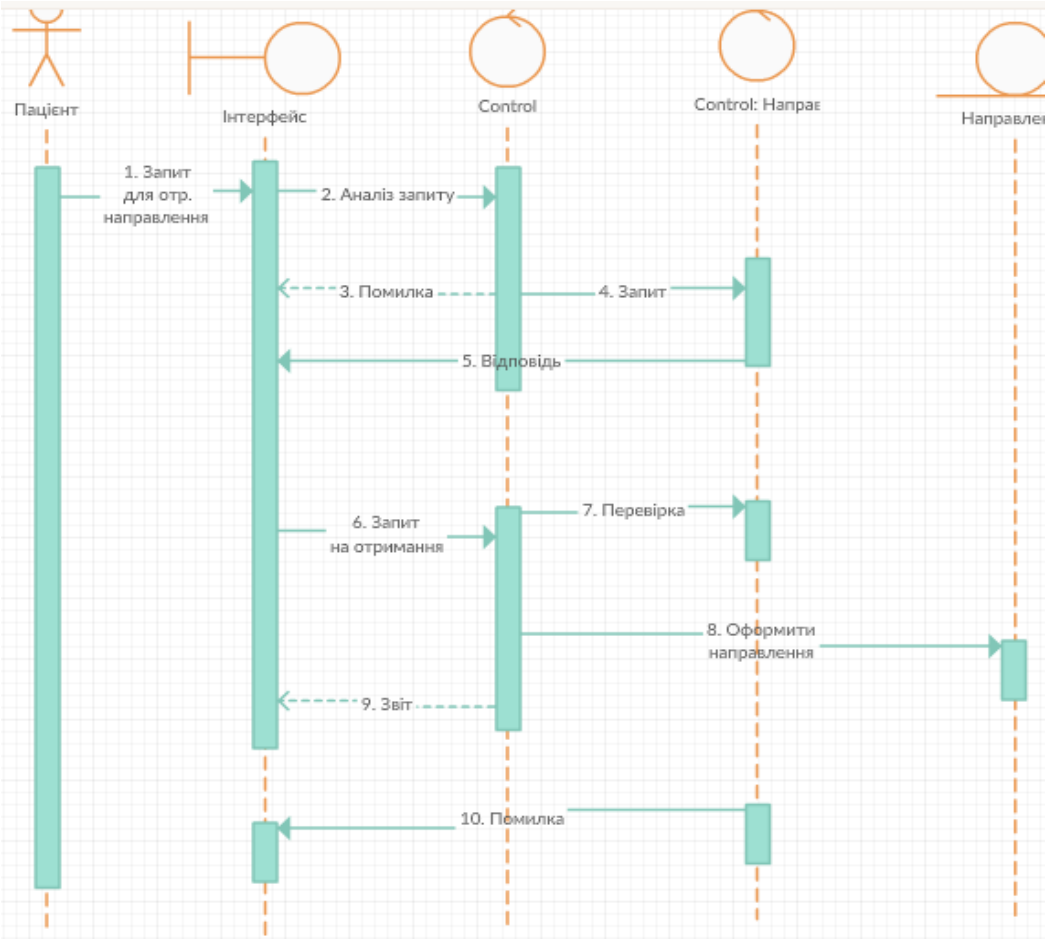


Рисунок 2.4 – Детальне проектування модуля «Оформлення направлення»

Детальне проектування, тобто сама розробка модулів застосунку - це діяльність, яка пов'язана з описом певних локальних обмежень проекту, зокрема шаблони проектування, патерни, архітектурні моделі, ідіоми програмування та рефакторингу системи. Згідно Intension/Locality Hyphotysis, основна відмінність між архітектурним та детальним проектуванням визначається локальним критерієм околиці (Locality Criteria), який стверджує, що дизайн програмного застосунку не є локальним дизайном, а є архітектурним дизайном саме тоді і тільки тоді, коли застосунок, який відповідає даному критерію може бути розширений в застосунок, яка йому це не задовольняє.

Модуль в програмуванні визначається, як функціонально закінчений фрагмент застосунку, оформлений у вигляді окремого набору файлів з вихідним програмним кодом, або іменованої безперервної певної його частини (наприклад,

Active Oberon), та призначений для його використання в інших застосунках, робота яких потребує його застосування.

Формування модулів дозволяє розбивати навіть складні завдання на менші згідно принципу модульності. Зазвичай це проектуються таким чином, щоб дати можливість програмістам досить зручну для повторного використання функціональність, деякий інтерфейс у вигляді певного набору типів, функцій, класів, констант, полів. Модулі можна об'єднуватися в пакети, сукупність класів, бібліотеки тощо.

Модулі бувають звичайними – які написані на тій же мові програмування, що і сам основна застосунок, де вони використовуються, або бути модулями розширення, що написані на мові, яка відмінна від основної мови застосунку. Модулі розширення зазвичай програмують на більш низькорівневій мові, яка дозволяє отримати суттєву перевагу в швидкості виконання та продуктивності усього застосунку.

Так, маючи п’ять окремих модулів-контролерів, застосунок можна розбити на п’ять окремих складових. Кожна частина якого відповідає за певний розділ веб-застосунку. Кожен контролер представляє собою деякий клас, що наслідує основний клас Controller та містить в собі методи, що відповідають за генерацію веб-сторінок веб-представлень. Кожен контролер взаємодіє з патерном об'єктно-реляційної поведінки Unit Of Work, задачею якого є відстеження змін об'єктів під час транзакцій, які відбуваються в нашій базі даних, та при роботі застосунку,як окремих фукцій та і всього застосунку

Все це відображено на діаграмі класів наведеній на рисунку 2.5 на якому зображено діаграму основних класів модуля Model. Так на діаграмі представлено наступні класи з їхніми членами:

- пацієнт;
- реєстрація;
- направлення;
- адміністратор.

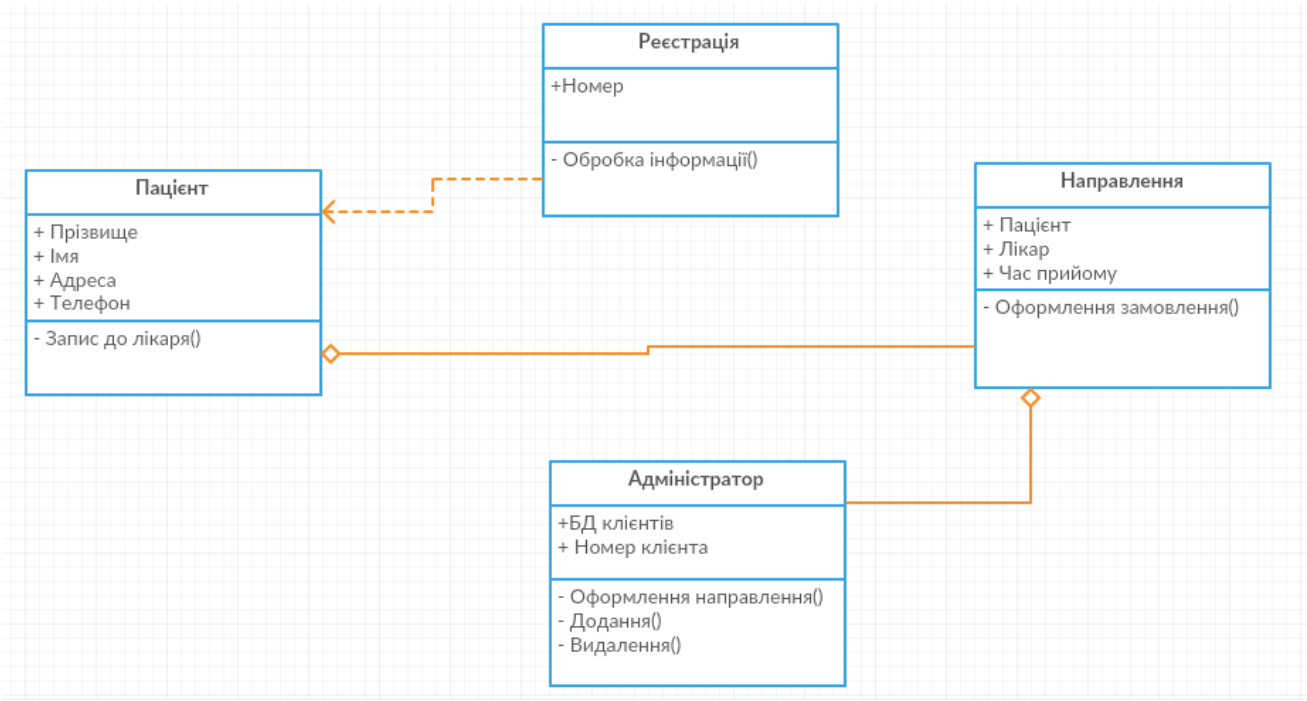


Рисунок 2.5– Діаграма класів

2.6 Детальне проектування даних

Загалом початок детального проектування даних має починатись з реалізації бази даних застосунку.

База даних використовується для введення-виведення даних, збереження, та обробки та видалення інформації. Бази даних володіють суттєвими перевагами відносно інших способів організованої певним чином інформації:

– для баз даних є характерним однократний введення та багатократне використання інформації, доступна інформація застосовується для вирішення багатьох завдань, забезпечується її багатоцільове і сумісне використання;

– бази даних існують та працюють незалежно від конкретних прикладних програм, що забезпечує уніфікацію для засобів організації даних і незалежність застосунків від організації даних;

– бази даних володіють модальністю (структурованістю, яка відображає певну предметну ділянку задачі);

– бази даних дозволяють встановити мінімально необхідний рівень надлишковості даних (тобто дані не дублюються при їх використанні різними користувачами);

– в базі даних забезпечується дотримання стандартів представлення даних, що спрощує їх створення та обслуговування;

– в базах даних забезпечується централізоване управління інформаційними ресурсами, синхронна підтримка даних для всіх прикладень, включаючи мови запитів і засоби захисту.

Використовуватиметься саме реляційна модель даних, адже вона забезпечує повну незалежність даних та має суворі правила проектування, які базуються на математичному апараті.

Обрано було саме СУБД MySQL через її високу продуктивність, низьку вартість. Також MySQL може використовуватися в середовищі багатьох різних систем UNIX, а також у середовищі Microsoft Windows.

Структура розробленої таблиці БД «Лікарі» показана на рисунку 2.6:

Рисунок 2.6 – Структура таблиці «Лікарі» в СКБД MySql

Структура таблиці БД «Прийом до лікаря» показана на рисунку 2.7:

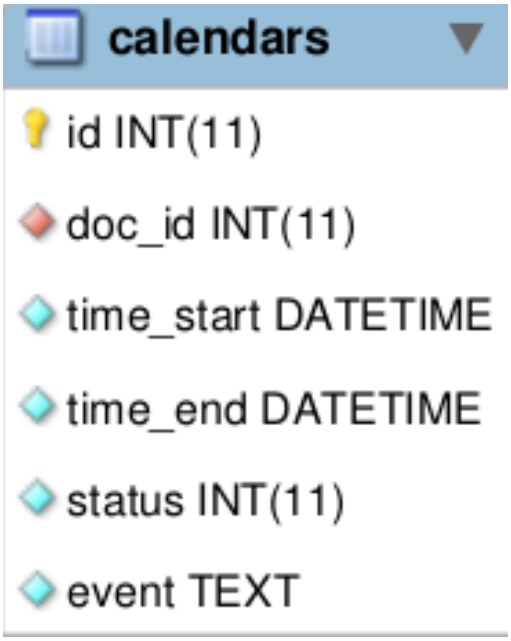


Рисунок 2.7 – Структура таблиці БД «Прийом лікаря» в СКБД MySql

Створюємо модель бази даних, та нормалізуємо її. Для приведення до другої нормальної форми, ми дані, що повторно з'являються в декількох рядках виносимо в окремі таблиці. Таблицю яка містила назву міста та її область розбиваємо на дві таблиці, на таблицю, яка буде містити індифікатор області, назву області та на таблицю яка буде містити назву міста, індифікатор міста, індифікатор області. Аналогічне робимо із іншими таблицями, які підлягають даній нормалізації.

Для того, щоб звести до третьої нормальної форми, потрібно винести поля, що залежать від основного ключа.

Тому для реалізації зв’язку ”many-to-many” створюємо додаткові проміжні таблиці “Місця роботи лікарів” та “місця роботи”, таким чином лікар може працювати в багатох лікарських закладах. Аналогічне робимо із іншими таблицями, які підходять даній нормалізації.

У результаті проведення нормалізації було отримано наступні таблиці: admins, calendars, certificates, cities, docspecs, doctors, docvnzes, docworkplaces, favorites, feedbacks, messages, newses, options, pages, posts,post_likes,prices, rating, regions, specs, subscriptions, telephones, users, user_settings, vnzes, workplaces

2.7 Висновок другого розділу

В даному розділі здійснено опис залежностей та опис інтерфейсу. Проведена декомпозиція на основі розробленої нами архітектури програмного застосунку, здійснено детальне проектування та розробка модулів застосунку модулів та детальне проектування даних.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Опис структури даних та моделі бази даних

Розглянемо основні модулі застосунку, які будуть розроблені у нашій системі. Першими будуть модулі для роботи із базою даних. Їх загальна кількість буде дев’ять та кожен із них, відповідно, буде відповідати за роботу із окремими відношенням в БД. Перерахуймо їх:

- сховище(repository) для роботи із відношенням Користувачі в БД;
- сховище(repository) для роботи із відношенням Логіни користувачів в БД;
- сховище(repository) для роботи із відношенням Пацієнти в БД;
- сховище(repository) для роботи із відношенням Лікарі в БД;
- сховище(repository) для роботи із відношенням Прийоми в БД;
- сховище(repository) для роботи із відношенням Консультації в БД;
- сховище(repository) для роботи із відношенням Коментарі в БД;
- сховище(repository) для роботи із відношенням Амбулаторії БД;
- сховище(repository) для роботи із відношенням Виписки в БД;
- сховище(repository) для роботи із відношенням Реєстратура в БД.

Кожен із наведених модулів буде реалізовувати CRUD-операції доступу до різних відношень в базі даних відповідно до встановленого шаблону «Об'єкт доступу до даних».

Після сховищ ідуть розроблені класи, які реалізовуються патерном Unit of Work. Цей патерн об’єднує сховища та надає ряд інструментів для роботи з базою даних. Наступним кроком йдуть контролери, які оброблюють запити користувачів.

Перерахуймо їх:

- контролер автентифікації користувача;
- контролер для роботи із користувацькими даними;
- контролер для роботи із даними лікарів;
- контролер для роботи із даними роботи амбулаторій;

– контролер для управління системою;

– контролер для керування аккаунтами.

Далі необхідно проаналізувати засоби розробки програмного продукту за допомогою яких можна реалізувати застосунок.

Визначимо структуру кожного модуля застосунку. Сховище для роботи із користувачами сайту відношення містить CRUD-операції для роботи із повязаими відношенням в базі даних. Дане сховище називається UserRepository. Дане сховище перевизначається з базової основи Identity Framework.

Модуль для роботи із відношенням запис на прийом в БД містить CRUD операції для роботи із цим відношенням в спроектованій базі даних. Даний модуль називається PrRepository.

Модуль для роботи із відношенням лікарі в БД містить CRUD операції для роботи із цим відношенням в спроектованій базі даних. Даний модуль називається DoctorRepository.

Модуль для роботи із відношенням амбулаторії у в БД містить CRUD-операції для роботи із цим відношенням в спроектованій базі даних. Даний модуль називається AmbRepository.

Модуль для роботи із відношенням управління в БД містить CRUD-операції для роботи із цим відношенням в спроектованій базі даних. Даний модуль матиме ім’я MangRepository.

Модуль для роботи із відношенням аккаунти в БД містить CRUD-операції для роботи із цим відношенням в спроектованій базі даних. Даний модуль матиме ім’я PostRepository.

Модуль для роботи із відношенням консультації в БД містить CRUD операції для роботи із цим відношенням в спроектованій базі даних. Даний модуль матиме ім’я ConsRepository.

Всі відношення, що наведені вище побудовані на базі шаблону Repository.

І на останок є група модулів що належать до контролерів. Архітектура цих модулів базується на базі архітектури Controller. Тому всі вони будуть мати схожу ідентичну будову.

3.2 Керівництво користувача

В результаті роботи створено веб-застосунок. При заході на сайт будь-якому користувачу потрібно зареєструватись, від чого в подальшому буде залежати функціонал роботи його в застосунку. Не авторизований користувач не може здійснювати працювати на сайті.

Якщо користувач зайшов вперше на сайт клініки та ще не є зареєстрованим в системі , то йому потрібно натиснути на кнопку «Створити акаунт». При натисканні кнопки «Увійти» користувач переходить на сторінку авторизації, де необхідно ввести логін та пароль. Вигляд форми авторизації можна на рисунку 3.1.

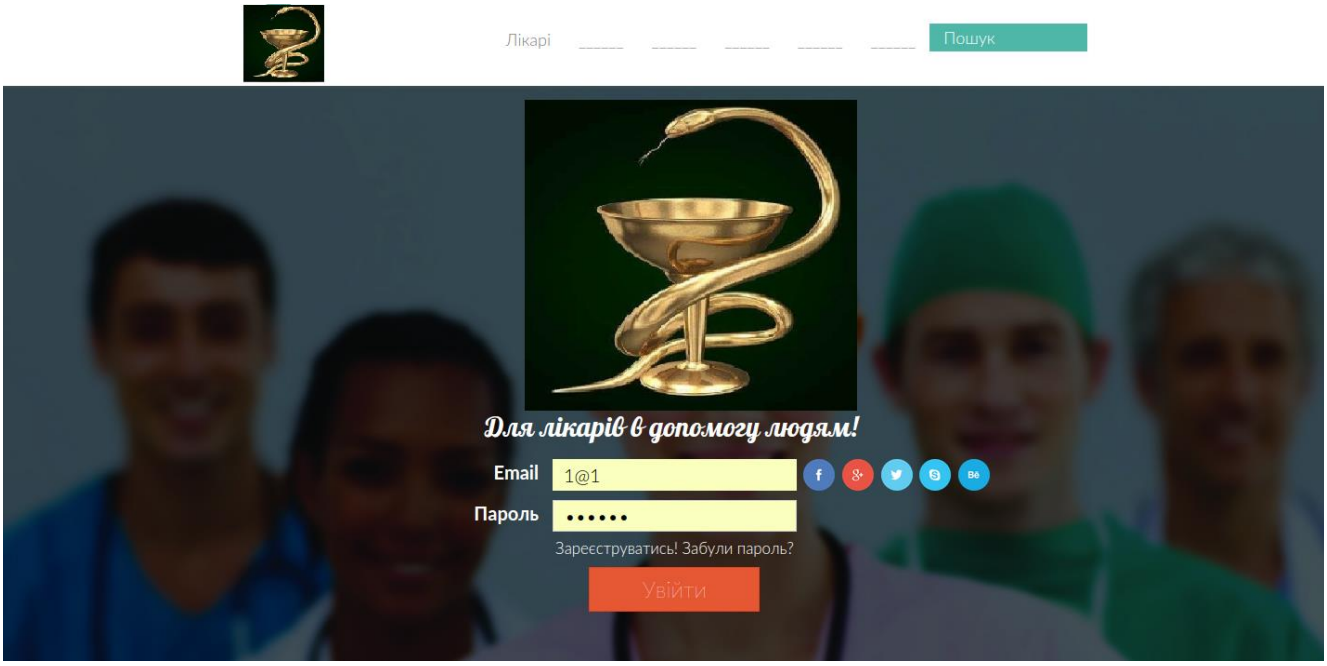


Рисунок 3.4 – Головна сторінка програми.

Після авторизації, користувач потрапляє на сторінку с вого профілю. , який зображено на рисунку 3.3.

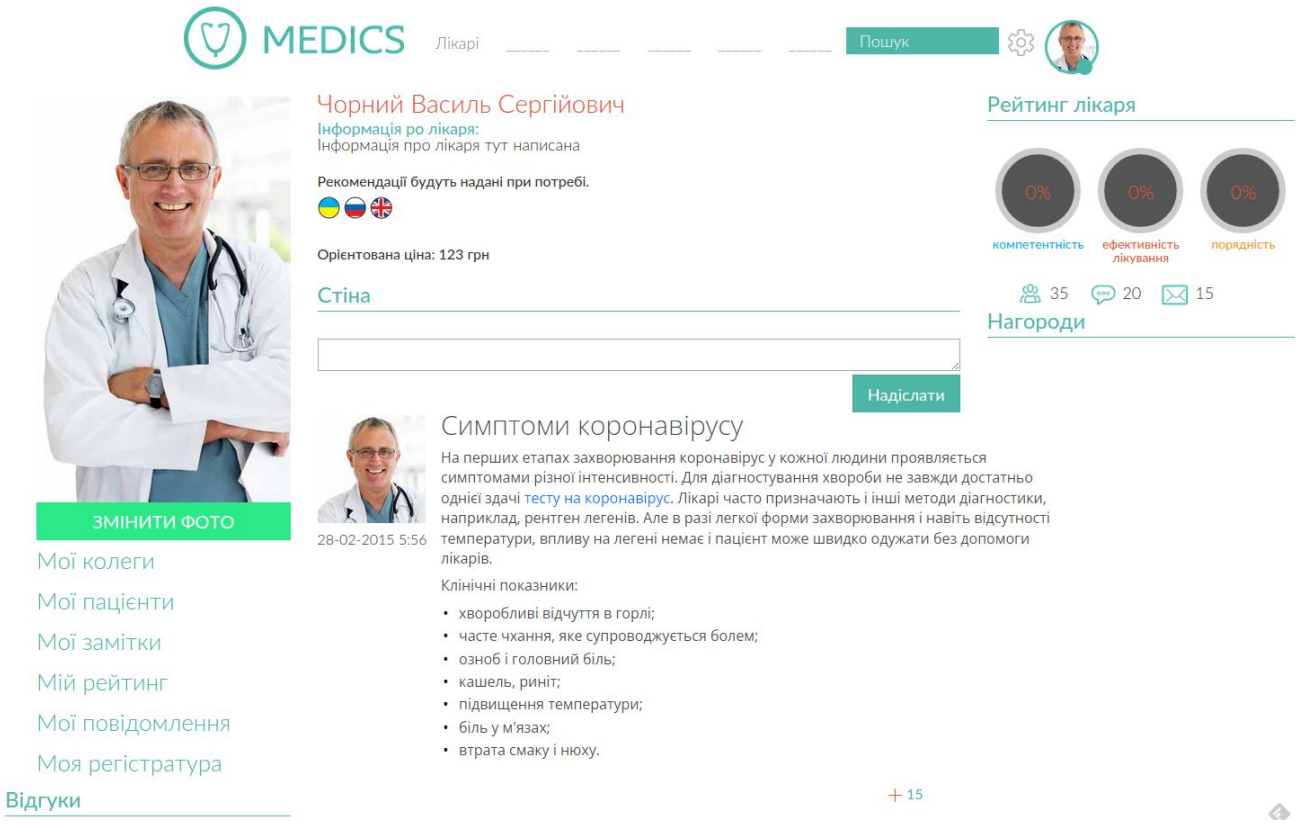


Рисунок 3.3 – Сторінка профілю

Перейшовши до лікаря на сторінку, користувач може записатися до нього на прийом (Рисунок 3.4):

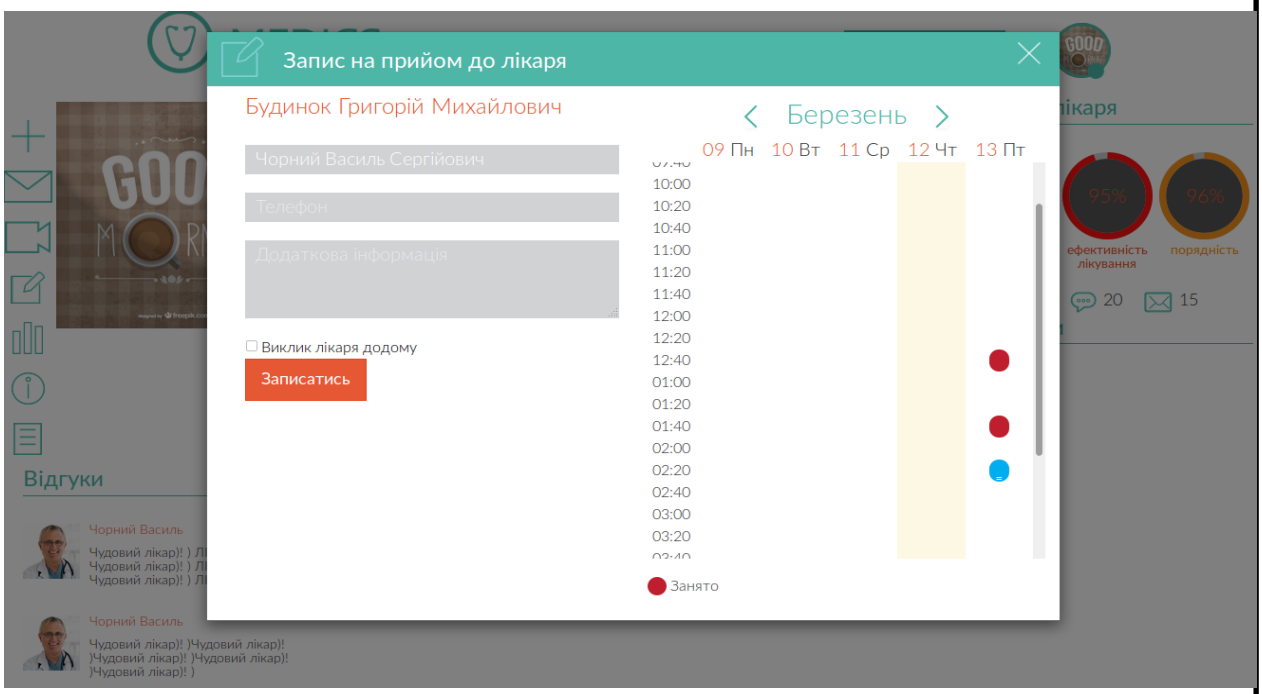


Рисунок 3.4 – Модальне вікно запису на прийом до лікаря

Користувацький досвід експлуатації сайту користувачами багато у чому залежить від інтерфейсу та його дизайну, розміщення елементів керування та кольорової гамми оформлення. Тому при виконанні застосунку нами були взяті до уваги основні вимоги до проектування інтерфейсів.

3.3 Налагодження та тестування системи

Для дослідження та аналізу розробленого веб-застосунку здійснемо процес його тестування.

Тестування програмного забезпечення в загальному включає аналіз, планування, проектування і виконання тестів застосунку.

Поняття Quality Assurance (QA) – це забезпечення якості застосунків. Тестувальник (QA-працівник) здійснює контроль, відповідає та забезпечує за якість роботи усіх програмних застосунків у компанії. Також він несе відповідальність за певні окремі етапи розробки застосунку: за вибір інструментів проектування та розробки, передбачає певні проблеми та нештатні ситуації, приймає участь участь у процесі вдосконалення застосунків тощо. В зону його відповідальності входять всі етапи розробки, починаючи з аналізу та опису проекту, супровід та тестування на етапі проектування, сам процес тестування готового продукту, та тестування в процесі експлуатації у разі внесення змін до застосунку.

Поняття Quality Control (QC) – це контроль якості застосунку. Основна задача спеціаліста QC, це перевірка певного конкретного застосунку, який включає в себе включає аналіз коду застосунку, його дизайну та сам процес тестування. QC-спеціаліст розробляє стратегію тестування для певного типу тестування застосунку, здійснює взаємодію з розробниками та здійснює саме тестування розроблювального застосунку.

Основна робота QA-працівника з тестування полягає у виконанні тестів над застосунком. Загалом, процес тестування, це перевірка результатів роботи програмного застосунку на відповідність наперед визначеним та заданим критеріям. QA-тестувальники займаються загалом тестуванням всього програмного застосунку та окремих його компонентів. Процес тестування відіграє досить важливу роль для забезпечення якості програмного застосунку.

Test Plan (TP) – План Тестування полягає у документі, в якому описується обсяг робіт із тестування застосунку, що починається з опису самого об'єкта, початок та закінчення тестування, стратегії тестування, розкладу тестування, критеріїв тестування, вимоги до обладнання необхідного в процесі роботи, спеціальних знань та також оцінки різного роду ризиків разом із можливими варіантами їх вирішення.

Test Design (TD) – тест дизайн, ще одни етап процесу тестування програмного застосунку, на якому здійснюється проектування та створення тестових випадків, так звані тест кейси, у відповідності до визначених раніше критеріїв якості та загальною метою тестування.

Test Case (TC) – Тестовий випадок, або «Тест кейс» – це документ, у якому описуються кроки, задані умови та параметри, які необхідні для перевірки та реалізації функції застосунку яка тестується, або деякої її частини.

Bug Report (BR) – звіт Помилка/Дефект – документ у якому описується ситуація та послідовність дій, яка привела до некоректної роботи застосунку чи деякої функції, із вказуванням причин появи та результату який очікувався (Expected Result).

Test Coverage(TC) – Тестове Покриття – це одна з метрик, яка здійснює оцінку якості процесу тестування застосунку. Вона передбачає із себе щільність покриття тестами вимог до виконуваного коду застосунку

Test Case Specification (TCS) – Деталізація Тест Кейсів — це визначений рівень деталізації з описом певних тестових кроків та необхідного результату

виконання, при якому забезпечується деяке визначене співвідношення часу проходження тесту до тестового покриття.

Test Case Pass Time (TCPT) – Час Проходження Тест Кейсу – визначений час від початку проходження визначених кроків тест-кейсу до моменту отримання результатів тестування програмного застосунку.

Враховуючи вище визначене створимо план тестування для розробки нашого програмного модуля запису на прийом до лікаря приватної клініки.

Метою складання даного ТР є опис процесу тестування нашого інтернет-сайту. Даний документ дозволяє отримати представлення про уявлення про заплановані роботи з тестування модуля.

Вихідними даними для тестування є програмний модуль запису на прийом до лікаря приватної клініки, який дозволяє користувачеві записатися на прийом до лікаря чи проведення аналізів.

Метою тестування програмного модуля запису на прийом до лікаря приватної клініки є перевірка роботи на правильність всіх його, та не функціональних можливостей у різних версіях веб-браузерів із використанням типових сценаріїв при його використанні. Частина виділеного часу (близько десяти відсотків) буде використана для здійснення тестування нетипових, потенційно спричинених помилок при роботи функцій застосунку та сценаріїв використання.

Підсумком процесу проведеного тестування будуть наступні документи:

– загальний висновок QA-тестувальників щодо загального стану застосунку, він дає користувачам, розробникам, програмістам та менеджерам даного застосунку загальне бачення коректності роботи сайту у різних типах браузерів;

– загальний звіт про результати тестування окремих функціональних частин застосунку та типові сценарії використання користувачами веб-переглядачів;

– задокументовані помилки та помилки у баг-трекері для розробників.

Тестування нашого застосунку проводитемо ручним методом з точки зору кінцевого користувача нашого веб-сайту.

Далі визначимо умови для тестування для нашого застосунку.

Наш програмний модуль повинен задовольняти потреби користувачів в різного роду активностях, які пов'язані із записом на прийом до лікаря та замовленнями послуг клініки.

Даля визначимо стратегію процесу тестування нашого програмного модуля.

Стратегія плану тестування є формальною, так як для побудови розширеного плану нам необхідно розуміти суть поточного стану проекту. В результаті першої ітерації запланованих функціональних тестів до нашого тест-плану внесемо зміни для його покращення. Перша ітерація функціональних тестів дасть нам чітке представлення про рівень відмовостійкості та стабільності системи та певним чином визначений набір тест-кейсів, що будуть виконані для в кожної конфігурації.

Такий підхід дасть нам змогу отримати досить детальний звіт щодо застосунку, який ми тестуємо та зосередити максимально нашу увагу на усіх виявлених вузьких місцях.

Замовнику надаватимуться постійно звіти про хід нашого тестування, зокрема про знайдені дефекти та пропозиції щодо покращення роботи модуля та оформлення його інтерфейсу і дизайну. Усі виявлені помилки дефекти заносяться також до баг-трекера замовника у вигляді фалів, які використовуються для необхідних кроків щодо їх виправлення.

Загалом нами було заплановано п’ять етапів проведення процесу тестування розробленого веб-сайту:

– на першому етапі здійснюється аналіз ТЗ, складається тест-план та здійснюється часткове тестування функціональних тестів;

– на другому етапі застосовується до детальному тестування функціональних тестів у якому здійснюється виявлення та описом помилок та дефектів;

– третій етап передбачає проведення тестування роботи різних браузерів з описом виявлених помилок та багів;

– а четвертому етапі здійснюється перевірка усунених розробниками помилок та проведення регресійного тестування;

– на п'ятому етапу здійснюється тестування дизайну та інтерфейсу з описом знайдених недоліків та помилок.

При такому плані тестування, досягається максимальна деталізація усієї глибини проведеного тестування застосунку, дозволяє нам більш детально та точніше визначити різні ресурси на затрати та дозволяє розробникам сайту виправляти помилки та неточності на якомога ранніх етапах.

Визначимо операційні системи для нашого тестування:

– Microsoft Windows 8;
– Microsoft Windows 10;
– Microsoft Windows 11;
– Android 7-12;
– iOS 16.

Визначимо операційні системи для нашого тестування:

– Edge;
– Google Chrome;
– Firefox;
– Opera;
– Safari.

Проводити тестування безпеки, а також стрес-тестування ми ми не будемо з врахуванням не досить великої їх доцільності для цього проекту та не через брак часу на його проведення.

Визанчимо та опишемо типи тестування

3.3.1 Функціональне тестування

Мета:

Виявлення різного роду функціональних помилок та невідповідностей технічному завданню. Очікувань від користувача шляхом здійснення стандартних, та нестандартних тестових сценаріїв.

Опис процесу функціонального тестування:

– Авторизація та реєстрація користувача:
– Авторизація користувача;
– Реєстрація користувача;
– Вхід для анонімного користувача;
– Вхід під логіном адміністратора системи;
– Відновлення логіну/паролю;
– Редагування облікового запису користувача.

Тестування особистого кабінету користувача:

– створення анкети користувача
– редагування анкети користувача;
– видалення анкети користувача;
– статус користувача;
– статус затвердження заявки;
– статус підтвердження/ скасування заявки;
– вхід/вихід з особистого кабінету користувача;
– тестування зворотнього зв’язку;
– статус прикріплення вкладення до повідомлення;
– статус повідомлення.

Тестування функції пошуку на сайті:

– здійснення пошуку за розділами сайту;
– здійснення пошуку публікацій;
– здійснення поширення та пошуку статей у соцмережах.

Тестування відображення рисунків та фотографій:

– перегортання сторінок

– відображення списку рисунків та фото.

Тестування функції банерів:

– перегортання рисунків та зображень.

Тестування функцій коментарів:

– додавання коментарю;

– редагування коментаря;

– видалення коментаря.

Тестування функцій фільтрування:

– фільтрування за певними категоріями

– фільтрування по послугах

– фільтрування по лікарях.

Тестування функцій сортування

– сортування за категоріями;

– сортування за лікарями;

– сортування за послугами;

– сортування за цінами.

Тестування функцій оплати послуг:

– оформлення послуг.

Тестування функцій сторінки лікаря:

– перегляд інформації про лікаря

– перегляд фотографій

– додавання коментарю

– вибір послуг лікаря.

Тестування функцій кошика:

– додавання послуг до кошика;

– видалення послуг з кошика.

Тестування функцій кросбраузерності здійснюється з головною метою сайту, з метою перевірки коректності роботи сайту, загального вигляду інтерфейсу та дизайну сайту у різних типах браузерів.

Тестування у браузерах:

– Edge;
– Google Chrome;
– Firefox;
– Opera;
– Safari.

Регресійне тестування та перевірка виправлених помилок та дефектів.

Головна мета тестування будь-якого застосунку, це перевірка внесених змін, які зроблені розробниками для того, щоб бути впевненим, що виправлена версія програмного коду не містить різних багів та помилок на усіх протестованих функціях застосунку.

Загалом, у ході здійсненого нами регресійного тестування було здійснено наступні види тестів:

– тестування версії;
– тестування основного функціоналу:
– тестування допоміжного функціоналу;
– верифікаційне тестування.

Тестування функціоналу та дизайну інтерфейсу.

Метою такого тестування є перевірка відповідності інтерфейсу розробленій специфікації у ТЗ.

Опис процесу тестування:

– тестування форми реєстрації користувача;
– тестування повідомлень з користувачем;
– тестування особистого кабінету користувача;
– тестування головного меню сайту;
– тестування контекстного меню сайту;
– тестування сторінки сайту;
– тестування банерів;
– тестування гіперопосилань.

Результати формування плану робіт наведені у таблиці 3.1.

Таблиця 3.1 – План робіт тестування сайту клініки

Завдання	Час роботи	Дата початку	Дата закінчення
Формування плану тестування	8 год	10.05.2023	10.05.2023
Проведення тестування	8 год	17.05.2023	31.05.2023
Аналіз результатів тестування	6 год	24.05.2023	31.05.2023
Формування звітів	8 год	31.05.2023	31.05.2023

Кінцевим результатом проведення тестування є оформлений звіт з остаточними результатами процесу тестування застосунку з описаними помилок, дефектів, багів та відповідними рекомендаціями щодо вдосконалення застосунку з точки зору кінцевого користувача застосунку.

Тестування функціональних можливостей та операційної поведінки застосунку необхідне з метою перевірки відповідності запланованому фунціоналу та специфікаціям.

Тестування, при якому ігноруються внутрішні механізми розроблювального застосунку або його окремого компонента, концентрується на вхідних та вихідних даних, та отримання результату на введення даних користувачем та умови виконання різних сценаріїв.

Загалом, метою функціонального тестування є перевірка та відповідність розроблювального програмного застосунку потрібній функціональній специфікації, яка зазначається в ТЗ з проектування.

В результаті виконання функціонального тестування створюються та звані чеклисти – одного із фундаментальних інструментів тестування програмних застосунків. Чеклисти дозволяють завжди мати перелік пунктів тестування,

фіксувати результати тестування та здійснювати контроль тестування та вести статистику про стан застосунку на конкретному епапі розробки чи експлуатації.

Загальні правила складання чеклистів:

– кожен пункт тестування – це визначенна окремаоперація;

– всі пункти чекліста – атомарні та повні операції;

– в списку тестів, подібні між собою операції оформляють окремими пунктами тестування;

– кожен пункт тестування починається з іменника, або дієслова в невизначеній формі;

– чекліст та його окремі пункти тестування можуть бути різного рівня деталізації, це залежить від вимог замовника до звітності, рівня складності застосунку тощо.

Головною метою чеклиста виступає необхідність врахування якомога більшого списку дій для найбільш повної перевірки тестами застосунку. складаючи пункти слід дотримуватися уніфікованої форми.

Чеклист потрібен при тестуванні для:

– встановлення переліку типів тестових завдань певного застосунку;

– поділу завдання за рівнем кваліфікації;

– збереження звітності про результати тестування.

Чеклист містить інформацію про:

1. Список тестів.

– назви пунктів тестів (детально);

– інформацію про стан перевірки;

– примітки (за необхідністю).

2. Оточення перевірки:

– збірка, на якій проводилося тестування;

– тестове оточення (якщо є);

– інформацію про тестувальників (в випадку декількох тестувальників) .

3. Результат перевірки;

– подається у вигляді таблиці та може бути доповнений детальним описом перевірки з рекомендаціями.

Перевіримо функціонал веб-сайту та внесемо дані у таблицю.

Таблиця 3.2 – Чеклист перевірки програмного модуля запису на прийом до лікаря приватної клініки

Номер тесту	Назва тесту	Результат тестування	Примітки
1.	Перевірка відображення тексту на сайті	Успіх	
2.	Перевірка кольорової гами інтерфейсу	Пройдено	
3.	Перевірка елементів інтерфейсу (меню, кнопки) їхнього розташування	Невдача	
4.	Перевірка модальних вікон	Успіх	
5.	Перевірка банерів	Успіх	
6.	Перевірка гіперпосилань	Успіх	
7.	Перевірка переходу по меню на сторінки	Успіх	
8.	Перевірка випадкового натиснення різних елементів	Пройдено	
9.	Перевірка часу завантаження сторінок	Пройдено	
10.	Перевірка дій на сторінках	Успіх	
11.	Перевірка дій на модальних сторінках	Успіх	
12	Перевірка прокрутки сторінок	Успіх	

Таблиця 3.3 – Чек-лист перевірки фунціоналу програмного модуля запису на прийом до лікаря приватної клініки у браузерах

№	Назва тексту	Edge	Google Chrome	Firefox	Opera	Safari
1.	Перевірка реєстрації на сайті	Успіх	Успіх	Успіх	Успіх	Успіх

2.	Перевірка входу в кабінет	Успіх	Успіх	Успіх	Успіх	Успіх
3.	Перевірка зворотнього зв’язку (повідом-лення, пошта)	Успіх	Успіх	Успіх	Успіх	Успіх
4.	Перевірка пошуку на сайті	Успіх	Успіх	Успіх	Успіх	Успіх
5.	Перевірка роботи гіперпосилань	Успіх	Успіх	Успіх	Успіх	Успіх
6.	Перевірка масштабування	Успіх	Успіх	Успіх	Успіх	Невдача
7.	Перевірка завантаження файлів	Невдача	Невдача	Невдача	Невдача	Невдача
8.	Перевірка заповнення модальних форм	Успіх	Успіх	Успіх	Успіх	Успіх
9.	Перевірка завантаження веб-сторінок	Пройдено	Пройдено	Пройдено	Пройдено	Пройдено
10.	Перевірка запису до лікаря	Успіх	Успіх	Успіх	Успіх	Успіх
11.	Перевірка замовлення послуг	Успіх	Успіх	Успіх	Успіх	Успіх
12.	Перевірка переходу на соціальні мережі	Невдача	Невдача	Невдача	Невдача	Невдача

3.4. Висновки до третього розділу

У третьому розділі нами здійснено опис структури даних та опис моделі бази даних. Наведено інструкцію користувача та проведено функціональне тестування застосунку.

Висновки

В даній кваліфікаційній випускній роботі реалізовано програний застосунок автоматизованої системи управління приватною клінікою. Більшість функцій визначених на етапі проектування та визначених в технічному завданні вдалося реалізувати у повній мірі.

Для досягнення цього результату в ході випускної роботи нами було здійснено першому розділі дослідження предметної області. Здійснено аналіз та визначення видів, методів та змісту інформаційного забезпечення предметної області. Проведено аналіз та визначення видів, методів та змісту інформаційного забезпечення предметної області. Здійснено постановку задачі та окреслено перелік задач для реалізації.

В другому розділі здійснено опис залежностей та опис інтерфейсу. Проведена декомпозиція на основі розробленої нами архітектури програмного застосунку, здійснено детальне проектування та розробка модулів застосунку модулів та детальне проектування даних.

У третьому розділі нами здійснено опис структури даних та опис моделі бази даних. Наведено інструкцію користувача та проведено функціональне тестування застосунку.

Автоматизація в нашому застосунку реалізується за рахунок сучасних засобів введення-виведення інформації, та реалізації пошуку з допомогою фільтрів.

Переваги проекту:

- інтуїтивний інтерфейс користувача;
- вдало підібраний дизайн сайту з застосуванням сучасних підходів;
- зручне додавання, видалення, редагування та аналізу інформації;
- відсутність прив’язки до певної апаратної та програмної платформи;
- відносно невисокі системні вимоги.

До недоліків застосунку можна віднести:

– обов’язковий доступ до глобальної мережі Internet;

– необхідність користувача з роллю адміністратора для повного заповнення даних;

– не повністю автоматизована система, має шляхи до вдосконалення.

Застосунок виконує основні функції, поставлені на етапі проектування та в технічному завданні, незважаючи на наявність деяких певних недоліків та можливих незначних помилок. Так, як проект створювався в навчальних цілях, для демонстрації навиків проектування, розробки архітектури застосунку та веб-програмування, задачу можна вважати виконаною, а всі виявлені недоліки, поправки та різноманітні зауваження від користувачів веб-сайту будуть нами виправлені і виявленні в ході подальшої експлуатації веб-застосунку.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Дипломний проект: методичні вказівки щодо його виконання для студентів спеціальності 121 «Інженерія програмного забезпечення». Л.П. Бедратюк, Г. І. Радельчук, Ю.В. Форкун, О.М. Яшина. Хмельницький:ХНУ, 2020. 77 с.
2. Kevin Yank. Build Yours Owns Database Driven Web Site Using PHP and MySQL. Four Edition. 2019. p.512
3. Davey Shafik - The PHP Anthology. 101 Essential Tips, Tricks and Hacks, 2nd Edition. 2019. 247 p.
4. Байкова В.Д., Макаренко О.М., Патлійников М.О., "Курс технологій MySql сервера." за редакцією академіка АПН України Дмитриченкова А.М., Харків, 2018. 447 с.
5. Юрченко Б.Г. Access 20. Бази даних та додатки. К: "ДіаСофт", 2018. 512с.
6. Ящевський В.Д., Макарчук О.М., Палійський М.О. Практичний курс інформатики" за редакцією академіка АПН України Паламарчука В.М., Київ, 2019. 418 с.
7. ISO/IEC/IEEE 42010, Systems & software engineering. Architecture description.
8. ISO/IEC 2382:2015. Information technology. Vocabulary, Part 1: Terms & definitions.
9. Головна сторінка Medics.ua. URL: https:// Medics.ua/ (дата звернення: 15.03. 2023)
10. Медична система HELSI. URL: https://helsi.me/ (дата звернення: 16.03. 2023)
11. Головна сторінка «Комунальна 5-а міська поліклініка міста Львова». URL: https://poliklinika2.lviv.ua/ (дата звернення: 15.03. 2023)
12. Patrick Rauland. WooCommerce Explained: Your Step-by-Step Guide to WooCommerce. WooExperts Press, 2019. 346 p.

13. Lisa Sabin-Wilson. WooCommerce For Dummies. For Dummies, 2019. 416 p.

14. Brad Williams, David Damstra, Hal Stern. Professional WordPress: Design and Development. Wiley, 2021. 552 p.

15. Brad Williams, Jeff Starr, Lisa Sabin-Wilson. WordPress for Beginners 2023: A Visual Step-by-Step Guide to Mastering WordPress. O'Reilly Media, 2023. 576 стор.

16. 16. Robbert Ravensbergen. Building E-commerce Solutions with WooCommerce: Customize and extend WooCommerce for your online store. Packt Publishing, 2020. 470 p.

17. Lisa Sabin-Wilson. WordPress All-in-One For Dummies. For Dummies, 2022. 912 p.

18. Patrick Rauland, Raúl P. García. "WooCommerce Cookbook". O'Reilly Media, 2020. 384 p.

19. Yannick Lefebvre. WordPress Plugin Development Cookbook. Packt Publishing, 2019. 358 p.

20. Stephen Burge. WooCommerce Explained: Your Step-by-Step Guide to Building an E-commerce Store with WooCommerce". OSTraining, 2021. 352 p.

21. Stephanie Leary. WordPress for Web Developers: An Introduction for Web Professionals". O'Reilly Media, 2021. 260 p.

22. David Trounce. WooCommerce Made Easy: A Step-by-Step Guide to Setting Up an Online Store". CreateSpace Independent Publishing Platform, 2019. 224 p.

23. Sufyan bin Uzayr. Mastering WooCommerce: Build a Professional Online Store with WordPress and WooCommerce. Packt Publishing, 2020. 320 p.

24. Karol Krol. WordPress 5 Complete: Build beautiful and feature-rich websites from scratch. Packt Publishing, 2018. 616 p.

25. Brad Williams, David Damstra, Hal Stern. Professional WordPress: Design and Development. Wiley, 2021. 552 p.

26. Eric Schwartzman. WooCommerce Tutorial: Build an E-commerce Website. Independently published, 2022. 152 p.

27. Vladimir Prelovac. WordPress Plugin Development: Beginner's Guide. Packt Publishing, 2019. 208 p.

28. Paul Thewlis. WordPress for Business Bloggers. Apress, 2022. 342 p.

29. Florian Kiersch. WooCommerce Explained: Master WooCommerce Plugin: A Step-by-Step Tutorial. Independently published, 2020. (208 стор.)

30. Patrick Rauland. WooCommerce Explained: Your Step-by-Step Guide to WooCommerce. WooExperts Press, 2021. 346 p.

31. Lisa Sabin-Wilson. WooCommerce For Dummies. For Dummies, 2019. 416 p.

32. Simon Brown. Software Architecture for Developers. Leanpub, 2019, 408 p.

33. Eric Evans. Domain-Driven Design Reference: Definitions and Pattern Summaries. Leanpub, 2019, 112 p.

34. Brendan Burns, Jeff Dean, Joe Beda. Designing Distributed Systems: Patterns and Paradigms for Scalable, Reliable Services. O'Reilly Media, 2019, 280 p.

35. Sam Newman. Monolith to Microservices: Evolutionary Patterns to Transform Your Monolith. O'Reilly Media, 2019, 386 p.

36. Microsoft Azure. Software Architecture Guide. Microsoft Press, 2019, 229 p.

37. Neal Ford, Rebecca Parsons, Patrick Kua. Building Evolutionary Architectures: Support Constant Change. O'Reilly Media, 2017, 206 p.

38. Cornelia Davis. Cloud Native Patterns: Designing Change-Tolerant Software. O'Reilly Media, 2020, 466 p.

39. Hector Garcia-Molina, Jeffrey D. Ullman, Jennifer Widom. Database Systems: The Complete Book.: Pearson, 2020, 1440 p.

40. Raghu Ramakrishnan, Johannes Gehrke. Database Management Systems.: McGraw-Hill Education, 2020, 1360 p.

41. Anthony Molinaro - SQL Cookbook: Query Solutions and Techniques for Database Developers.: O'Reilly Media, 2019, 576 p.

42. Martin Kleppmann - Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems.: O'Reilly Media, 2017, 614 p.

43. Pramod J. Sadalage, Martin Fowler. NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence. Addison-Wesley Professional, 2012, 192 p.

44. Alex Petrov. Database Internals. A Deep Dive into How Distributed Data Systems Work.O'Reilly Media, 2020, 402 p.

45. S. K. Singh. Database Systems: Concepts, Design and Applications.: Pearson, 2020, 824 p.

46. Anthony DeBarros.Practical SQL: A Beginner's Guide to Storytelling with Data. No Starch Press, 2018, 504 p.

47. Seyed M. M. Learning MySQL: Get a Handle on Your Data. O'Reilly Media, 2020, 472 p.

48. Nathan Marz, James Warren. Data Management at Scale: Lessons Learned from Building Large Distributed Systems. O'Reilly Media, 2020, 200 p.

49. Shannon Bradshaw, Kristina Chodorow. MongoDB: The Definitive Guide: Powerful and Scalable Data Storage. O'Reilly Media, 2013, 448 p.

50. Viktor Mayer-Schönberger, Kenneth Cukier. Big Data: A Revolution That Will Transform How We Live, Work, and Think. Eamon Dolan/Houghton Mifflin Harcourt, 2013, 256 p.

51. 53. Michael Kifer, Arthur Bernstein, Philip M. Lewis, Leonidas Fegaras. Database Systems: An Application-Oriented Approach. Pearson, 2021, 1000 p.

52. Baron Schwartz, Peter Zaitsev, Vadim Tkachenko. High Performance MySQL: Optimization, Backups, and Replication. O'Reilly Media, 2012, 828 p.

ДОДАТОК А
(обов’язковий)

ТЕХНІЧНЕ ЗАВДАННЯ

Введення

Робота виконується в рамках проекту розробки веб застосунку для автоматизації клієнто-орієнтованих бізнес-процесів приватної клініки. Технічне завдання розроблено у відповідності до стандарту ДСТУ 7363:2013.

1 Підстава для розробки

Підставою для розробки є «Завдання на кваліфікаційну роботу», затверджене завідувачем кафедри інженерії програмного забезпечення. Найменування розробки: Програмний модуль запису на прийом до лікаря приватної клініки.

2 Призначення розробки

2.1 Функціональне призначення

Функціональним призначенням додатку є клієнтська частина для перегляду інформації про запис на прийом до лікаря приватної клініки, та серверна частина з можливістю редагування вмісту сайту.

2.2 Експлуатаційне призначення

Програма повинна експлуатуватися на будь-яких пристроях, на яких є браузер. Кінцевим користувачем додатку може виступати будь-яка особа.

3 Вимоги до програми

3.1 Вимоги до функціональних характеристик

Для звичайного користувача:

- перегляд інформаційних блоків про компанію;
- перегляд лікарів;
- перегляд прийомів та консультацій лікарів;
- окремі сторінки під різні будівлі з їх наявними плануваннями та статусами будови;
- можливість звернутись онлайн до реєстратури;
- можливість переглянути потрфоліо лікарів;
- можливість конфігурації готового інтер’єру квартири;
- можливість запису на прийом;
- можливість замовлення послуг клініки.

Для адміністратора:

- авторизація;
- створення ролей адміністраторів з відповідними доступами до редагування вмісту сайту;
- можливість створення та редагування блоків про ліакрів;
- можливість додавання об’єктів у портфоліо лікарі;
- можливість завантаження фото/відео на веб-сайті;
- можливість створення та редагування окремих сторінок лікаря з необхідною інформацією;
- можливість оновлювати статуси реєстратури;
- можливість створювати контент працівниками клініки;
- можливість опрацювання усіх вхідних онлайн-звернень з веб-сайту;
- можливість загрузки та оновлення необхідних документів;
- система обліку усіх вхідних заявок;
- можливість редагування усього контенту сайту.

3.2 Вимоги до надійності

Веб-додаток повинен забезпечувати такі вимоги до надійності:

- обробляти невірні дії користувача і попереджати його про можливі наслідки;
- можливість самостійно відновлюватись у разі збою;
- можливість резервного копіювання бази даних.

3.3 Умови експлуатації та вимоги до технічних засобів

Веб-додаток повинен працювати на всіх пристроях, які мають браузер та стабільний доступ до мережі «Інтернет»: смартфонах, планшетах та комп’ютерах. Браузер може бути будь-яким: Edge, Safari, GoogleChrome, Opera, MozillaFirefox, тощо.

Для роботи платформи без збоїв, потрібно, щоб пристрій, на якому вона запускається задовольняв наступні мінімальні вимоги:

- 100 Гб внутрішньої пам’яті;

- 2 Гб оперативної пам’яті;
- 2-ядерний процесор;
- доступ до мережі «Інтернет» зі швидкістю мінімум 10 Мбіт/с.

3.4 Вимоги до інформаційної та програмної сумісності

Для розробки веб-застосунку було використаний фреймворк WordPress, фронтенд JavaScript фреймворк Vue.js, бекенд PHP, та база даних MySQL.

3.5 Спеціальні вимоги

Програма повинна мати зручний та зрозумілий зовнішній інтерфейс користувача.

4 Вимоги до програмної документації

У момент здачі проекту замовнику надається наступний набір документів:

- текст програми;
- опис програми;
- технічне завдання;
- керівництво користувача.

5 Стадії та етапи розробки

Стадії та етапи розробки веб-застосунку для автоматизації клієнто-орієнтованих бізнес-процесів приватної клініки А.1.

Таблиця А.1 – Стадії та етапи розробки проекту

Стадія розробки	Етапи робіт	Зміст робіт
1	2	3
Технічне завдання 02.01.23 – 31.01.23	Обґрунтування необхідності розробки програми	Коротка характеристика програмного забезпечення; підстава і призначення розробки; вимоги до програмної системи і документація; стадії і етапи розробки програми; порядок контролю і приймання
Ескізний проект 01.02.23 – 26.02.23	Розробка ескізного проекту	Попередня розробка структури вхідних і вихідних даних; уточнення середовища програмування; розробка і опис загальної алгоритмічної структури

Кінець таблиці А.1

1	2	3
Технічний проект 29.02.23 – 19.03.23	Розробка технічного проекту	Уточнення структури вхідних і вихідних даних; розробка

		докладного алгоритму; розробка структури програми
Робочий проект 20.03.23 – 15.04.23	Розробка програмного забезпечення	Реалізація програмного забезпечення; відлагодження; проведення попереднього тестування
Розробка програмної документації 16.04.23 – 22.04.23	Розробка документації до програмного забезпечення	Розробка необхідної документації, передбаченої технічним завданням
Тестування системи 23.04.23 – 30.04.23	Проведення тестування програмного забезпечення	Розробка методики тестування; проведення основних тестів; коректування програмного забезпечення
Впровадження	Підготовка і передача програми	Підготовка і розгортання програмного забезпечення
Розробка програмної документації 16.04.23 – 22.04.23	Розробка документації до програмного забезпечення	Розробка необхідної документації, передбаченої технічним завданням

6 Порядок контролю та приймання

Контроль здійснюється кінцевими користувачами системи, підключеними на етапі тестування додатку.

ДОДАТОК Б
(обов’язковий)

ДІАГРАМИ

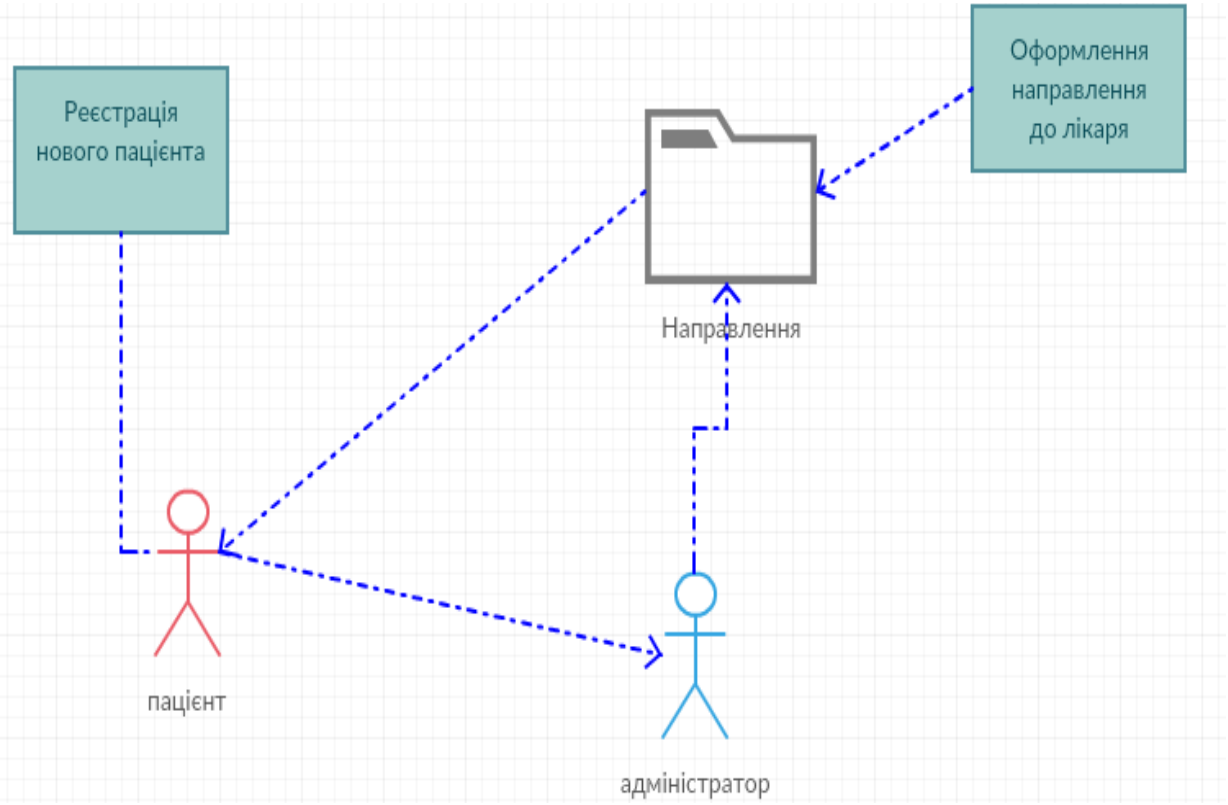


Рисунок Б.2 - Опис інтерфейсів програмного модуля

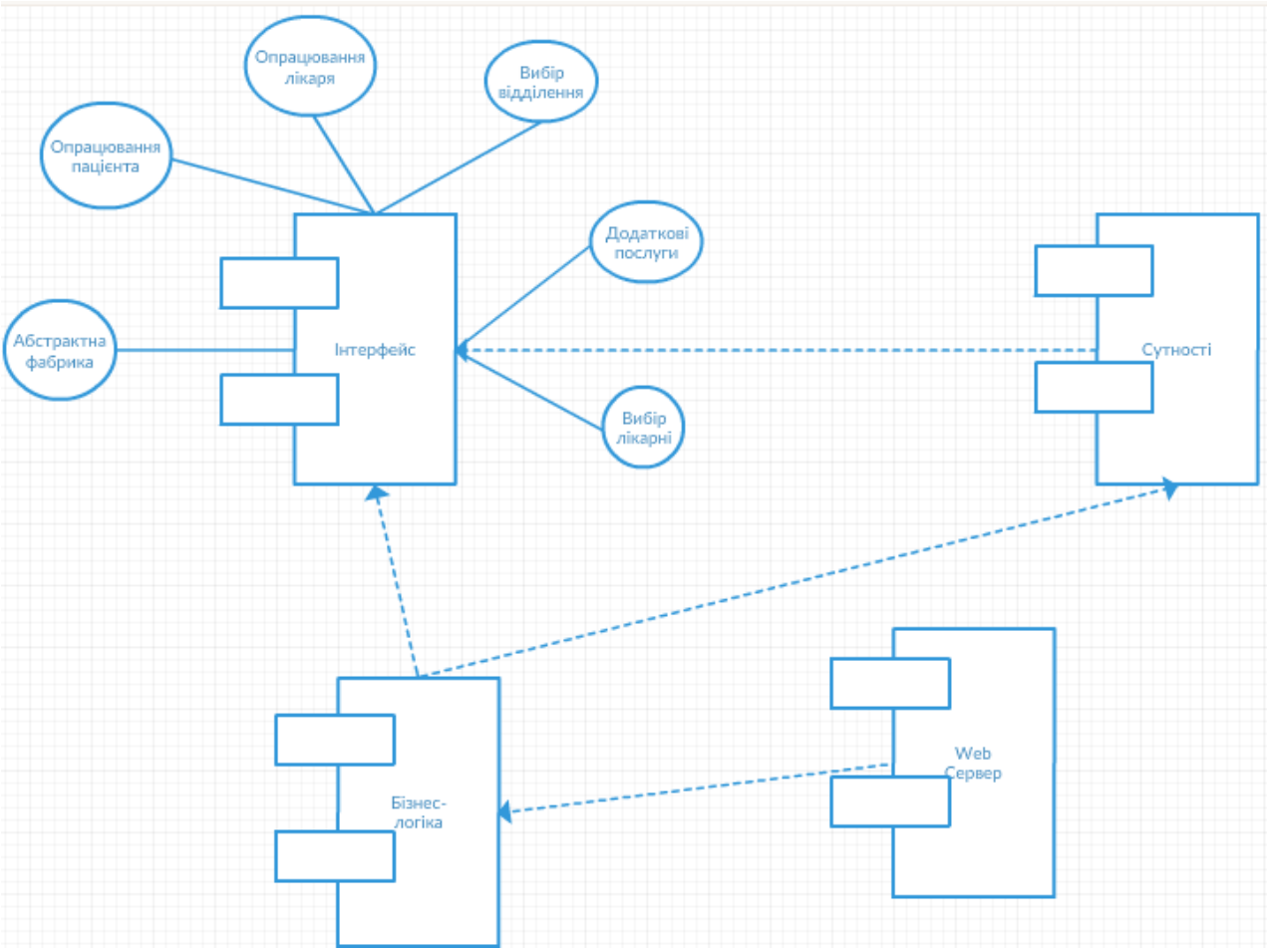


Рисунок Б.2 - Опис інтерфейсів програмного модуля

ДОДАТОК В
(обов’язковий)

КОД (ЛІСТИНГ) ПРОГРАМИ

```
<?php
namespace App\Controller;
class Doc extends \App\Page
{


    public function action_index()
    {
        $this->view->subview = 'doc';
        $this->view->doc = $this->pixie->db
            ->query('select')
            ->table('doctors')
            ->join('users', array('users.id', 'doctors.id'), 'left')
            ->execute();
        if ($this->pixie->auth->user()) {
            if ($this->pixie->auth->user()->id == $id) {
                $this->view->iam = true;
            } else $this->view->iam = false;
        }
    }


    public function action_id()
    {
        $this->view->title = "Page title";
        $this->view->iam = false;
        $id = $this->request->param('id');
        $this->view->id = $this->request->param('id');
        if ($this->pixie->auth->user()) {
            if ($this->pixie->auth->user()->id == $id) {
                $this->view->iam = true;
            } else $this->view->iam = false;
        }
        $this->view->doc_id = $id;
        if ($this->pixie->auth->user() && $this->view->iam) {
            $me = $this->pixie->auth->user()->id;
            $this->view->count_messages = $this->pixie->orm
                ->get('message')
                ->where('user_to', $me)
                ->where('and', array('status', 0))
                ->count_all();


        }
        $this->view->feedbacks = $this->pixie->orm->get('feedback')-
>where('into_doc', $id)->order_by('date', 'desc')->find_all()-
>as_array();
        $competence = 0;
        $effectiveness = 0;
        $honesty = 0;
        $i = 0;
        foreach ($this->view->feedbacks as $one) {
            $competence += $one->competence;
```

```
            $effectiveness += $one->effectiveness;
            $honesty += $one->honesty;
            $i++;
        }
        if ($i == 0) $i = 1;
        $this->view->competence = round($competence / $i);
        $this->view->effectiveness = round($effectiveness / $i);
        $this->view->honesty = round($honesty / $i);
        $this->view->news = $this->pixie->orm->get('news')->find_all();
        $this->view->posts = $this->pixie->orm->get('post')-
>where('user', $id)->order_by('date', 'desc')->find_all()->as_array();
        // $this->view->posts =$this->pixie->orm->get('post_like')-
>where('user', $id)->find_all()->as_array();
        $this->pixie->router->get('default')->url(array('id' => 5));
        $this->view->doc = $this->pixie->db
            ->query('select')
            ->table('doctors')
            ->join('users', array('users.id', 'doctors.id'), 'left')
            ->where('id', $id)
            ->execute();
        function calculate_age($birthday)
        {
            $birthday_timestamp = strtotime($birthday);
            $age = date('Y') - date('Y', $birthday_timestamp);
            if (date('md', $birthday_timestamp) > date('md')) {
                $age--;
            }
            return $age;
        }
        $this->view->cal = $this->pixie->orm->get('calendar')-
>where('doc_id', $id)->find_all()->as_array();
        foreach ($this->view->doc as $this->view->doctor) {
            $this->view->old = calculate_age($this->view->doctor-
>b_date);
            $this->view->exp = calculate_age($this->view->doctor->exp);


        }


        $this->view->subview = 'doc_view';
    }


    public function action_view()
    {


        $this->view->subview = 'doc';
        $id = $this->request->param('id');
```

```
        $this->view->doc = $this->pixie->db
            ->query('select')
            ->table('doctors')
            ->join('users', array('users.id', 'doctors.id'), 'left')
            ->where('id', $id)
            ->execute();




    }


}
```

Пошук:

```
<?php
/**
 * Created by PhpStorm.
 * User: amidyshka
 * Date: 09.03.15
 * Time: 22:54
 */


namespace App\Controller;




use App\Page;


class Docs extends Page
{


    public function action_index()
    {
        $region = $this->request->get('region');
        $qcity = $this->request->get('city');
        $minCost = $this->request->get('minCost');
        $maxCost = $this->request->get('maxCost');
        if ($this->pixie->auth->user() && !isset ($region)) $this-
>redirect('/docs?region=' . $this->pixie->auth->user()->region . '&city='
. $this->pixie->auth->user()->city);
        $this->view->allspecs = $this->pixie->orm->get('spec')-
>find_all()->as_array();
        $this->view->docspecs = $this->pixie->orm->get('docspec')-
>find_all()->as_array();
        $this->view->cities = $this->pixie->orm->get('city')->find_all()-
```

```
>as_array();
        $this->view->regions = $this->pixie->orm->get('region')-
>find_all();
        $this->view->region = $this->request->get('region');


        $this->view->qcity = $this->request->get('city');
        $this->view->allwork = $this->pixie->orm->get('workplace')-
>find_all()->as_array();


        $this->view->subview = 'docs';
        $spec = $this->request->get('spec');
        $this->view->spec = $spec;
        $work = $this->request->get('work');
        $this->view->work = $work;
        $city = $this->request->get('city');
        $search = $this->request->get('search');




        $docs = $this->pixie->db
            ->query('select')
            ->table('doctors')
            ->join('users', array('users.id', 'doctors.id'), 'left')
            ->join('docspecs', array('docspecs.doc', 'doctors.id'),
'left')
            ->join('docworkplaces', array('docworkplaces.doc_id',
'doctors.id'), 'left');


        $search = preg_replace('/\s{3,}/', ' ', $search);
        $search = trim($search);
        $words = explode(" ", $search);
        $this->view->search = $search;
        $this->view->ss = false;
        if (isset($spec) && ($spec != "all"))
            $docs->where('and', array('docspecs.spec', $spec));
        if (isset($work) && ($work != "all"))
            $docs->where('and', array('docworkplaces.workplace_id',
$work));
        if (isset($city) && ($city != "all"))
            $docs->where('and', array('users.city', $city));
        if (isset($minCost) && isset($maxCost))
            $docs->where(array(
                array('price','>',$minCost),
                array('price','<',$maxCost)));
        if (strlen($search) > 3) {


            $docs->where(array(
                array('and', array(
```

```
                    array('users.sname', 'LIKE', '%' . $words[0] . '%'),


                )
                )
            ));
            foreach ($words as $word)
                $docs->where('or', array('users.name', 'LIKE', '%' .
$word . '%'));




        }




        $arr = $docs->execute()->as_array();


        $new_array = array();
        foreach ($arr as $item)
            if (!array_key_exists($item->doc, $new_array))
                $new_array[$item->doc] = $item;


        $this->view->docs = $new_array;


    }


}
```

Адміністрування:

```
<?php
/**
 * Created by PhpStorm.
 * User: amidyshka
 * Date: 05.03.15
 * Time: 5:16
 */


namespace App\Controller;




class Admin extends \App\Admin
{
    public function action_index()
    {
```

```
        if ($this->pixie->auth->service('admin')->user()) {
            $this->view->subview = 'index';
            $this->view->doc = $this->pixie->db
                ->query('select')
                ->table('doctors')
                ->join('users', array('users.id', 'doctors.id'), 'left')
                ->execute();
            $this->view->specs = $this->pixie->orm->get('docspec')-
>find_all()->as_array();
        } else {
            $this->redirect('/admin/login');
        }


    }


    public function action_login()
    {
        $this->view->subview = 'login';
        if ($this->pixie->auth->service('admin')->user()) {
            $this->redirect('/admin');
        }
        if ($this->request->method == "POST") {
            $login = $this->request->post('username');
            $password = $this->request->post('password');
            $auth = $this->pixie->auth->service('admin')-
>provider('password')
                ->login($login, $password);
            if ($auth) return $this->redirect('/admin');
            if (!$auth) return $this->redirect('/admin/login');
        }
    }


    public function action_logout()
    {
        $this->pixie->auth->service('admin')->logout();
        $this->redirect('/');
    }


    public function action_dash()
    {
        if ($this->pixie->auth->service('admin')->user()) {
            $this->view->subview = 'dash';
            $this->view->doctor = $this->pixie->orm->get('doctor')-
>find_all()->as_array();


        } else {
            $this->redirect('/admin/login');
        }
```

```
    }


    public function action_doc()
    {
        if ($this->pixie->auth->service('admin')->user()) {
            $this->view->subview = 'docset';
            $id = $this->request->param('id');
            $this->view->doc = $this->pixie->orm->get('doctor')-
>where('id', $id)->find();
            $this->view->cal = $this->pixie->orm->get('calendar')-
>where('doc_id', $id)->find_all()->as_array();
            $this->view->specs = $this->pixie->orm->get('docspec')-
>where('doc', $id)->find_all()->as_array();
            $this->view->works = $this->pixie->orm->get('docworkplace')-
>where('doc_id', $id)->find_all();
            $this->view->allspecs = $this->pixie->orm->get('spec')-
>find_all()->as_array();
            $this->view->allwork = $this->pixie->orm->get('workplace')-
>where('city', $this->view->doc->user->city)->find_all()->as_array();
            $this->view->cities = $this->pixie->orm->get('city')-
>find_all()->as_array();
            $this->view->regions = $this->pixie->orm->get('region')-
>find_all();
            $this->view->vnzs = $this->pixie->orm->get('vnz')-
>find_all();
            $this->view->region = $this->pixie->orm->get('city')-
>where('id', $this->view->doc->user->city)->find();


            if ($this->request->method == "POST") {
                $user = $this->pixie->orm->get('user', $this->view->doc-
>id);
                $user->name = $this->request->post('username');
                $user->sname = $this->request->post('firstname');
                $user->fname = $this->request->post('lastname');
                $user->city = $this->request->post('city');
                $user->region = $this->request->post('region');
                $user->save();
                $this->view->doc->id_vnz = $this->request->post('vnz');
                $this->view->doc->info = $this->request->post('info');
                $this->view->doc->save();
            }


        } else {
            $this->redirect('/admin/login');
        }


    }
```

```
    public function action_add_doc()
    {
        if ($this->pixie->auth->service('admin')->user()) {
            $this->view->subview = 'docadd';


            $this->view->options = $this->get_option();
            $this->view->regions = $this->pixie->orm->get('region')-
>find_all();
            $this->view->cities = $this->pixie->orm->get('city')-
>find_all();
            $this->view->vnzs = $this->pixie->orm->get('vnz')-
>find_all();
            $this->view->specs = $this->pixie->orm->get('spec')-
>find_all();




            $this->view->user = $this->pixie->auth->user();


            if ($this->request->method == 'POST') {


                $date = (string)$this->request->post('year') . '-' .
(string)$this->request->post('month') . '-' . (string)$this->request-
>post('day');
                $gen = '';
                if ($this->request->post('gender') == 'Чоловіча') $gen =
1;
                if ($this->request->post('gender') == 'Жіноча') $gen = 0;


                $password = $this->request->post('password');
                $hash = $this->pixie->auth->provider('password')-
>hash_password($password);


                $user = $this->pixie->orm->get("user");
                $user->sname = $this->request->post('sname');
                $user->name = $this->request->post('name');
                $user->fname = $this->request->post('fname');
                $user->email = $this->request->post('email');
                $user->region = $this->request->post('region');
                $user->city = $this->request->post('city');
                $user->b_date = $date;
                $user->password = $hash;
                $user->gender = $gen;
                $user->save();
                //echo $doc_now;
```

```
                $expdate = (string)$this->request->post('w_year') . '-' .
(string)$this->request->post('w_month') . '-' . (string)$this->request-
>post('w_day');
                $doc_now = $user->id;
                $doctor = $this->pixie->orm->get("doctor");
                $doctor->id = $doc_now;
                $doctor->info = $this->request->post('info');
                $doctor->id_vnz = $this->request->post('vnz');


                $doctor->exp = $expdate;
                $doctor->save();




                $doc_vnz = $this->pixie->orm->get("doc_vnz");
                $doc_vnz->doc = $doc_now;
                $doc_vnz->vnz = $this->request->post('vnz');
                $doc_vnz->save();


                $this->redirect('/admin/doc' . $doc_now);
            }


        } else {
            $this->redirect('/admin/login');
        }
    }
```

ДОДАТОК Г
(обов’язковий)

ПРЕЗЕНТАЦІЙНІ МАТЕРІАЛИ

Хмельницький Національний Університет
Факультет інформаційних технологій

Кваліфікаційна робота

Тема: Програмний модуль запису на прийом до лікаря приватної клініки

Студент: Тропарчук А. О.
Керівник: Форкун Ю. В.

Метою створення програмного модуля та інтерфейсу є:

- Розробка архітектури створення автоматизованої системи управління лікувальним закладом
- Автоматизація робочого місця для працівників лікарні
- Збільшення безпеки збереження даних
- Зменшення часу на оформлення запису на прийом до лікаря

Постановка задачі та перелік задач для реалізації

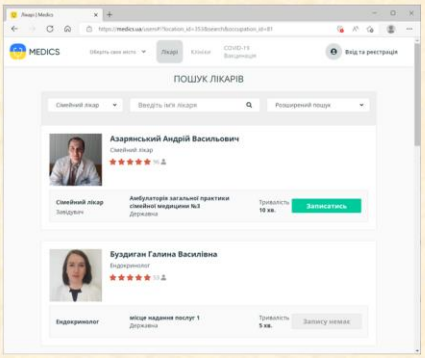
- авторизація адміністратора та реєстратора в програмі;
- інтуїтивний інтерфейс;
- сторінка адміністрування ресурсу;
- пошукова система, за заданими фільтрами;
- зручне створення замовлень прийому, їх редагування;
- перегляд та редагування даних;
- формування звітів в залежності від отриманих результатів;

Опис декомпозиції

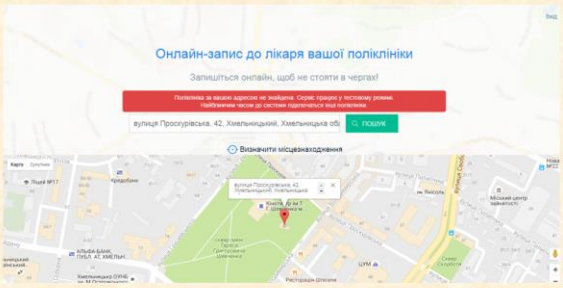
Поставлену задачу було розділено на такі модулі:

- модуль навігації, що включає в себе інтуїтивно розташовані головне меню, панелі інструментів та контекстні меню для кожного блоку редагування;
- модуль редагування даних містить усі необхідні елементи інтерфейсу, за допомогою яких можна додавати, редагувати та видаляти дані, призначений для зміни інформації у базі даних;
- модуль фільтрації дозволяє шукати інформацію відповідно до запитів користувача та призначений для наглядного відображення даних;
- модуль перегляду включає в себе таблиці та поля з інформацією, графік роботи лікаря.

Аналіз досліджень та існуючих рішень

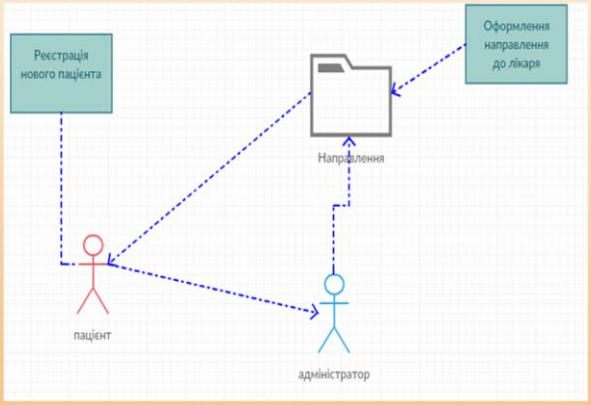


сайт «medics.ua»



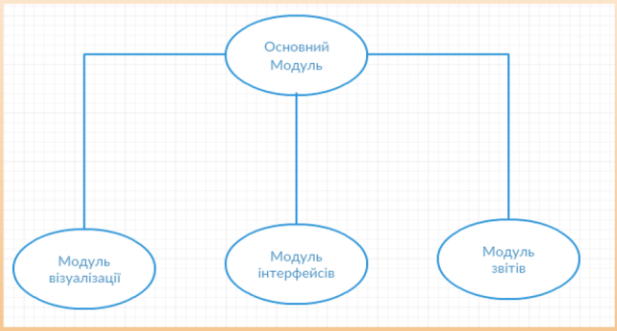
сайт «reestratura.com»

Опис залежностей



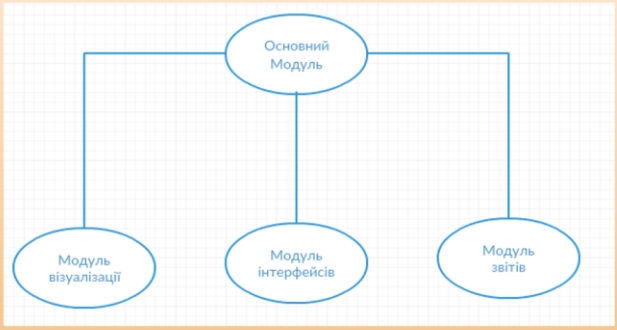
Залежність між компонентами

Опис залежностей



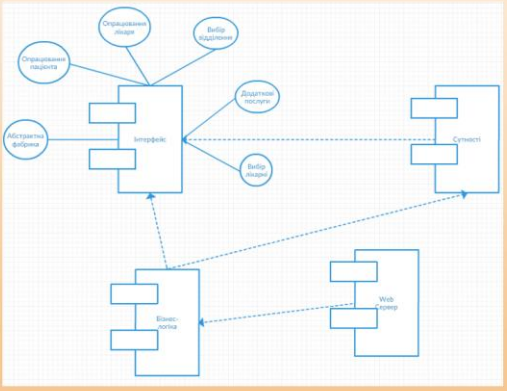
Залежність між модулями пакету Пацієнт

Опис залежностей



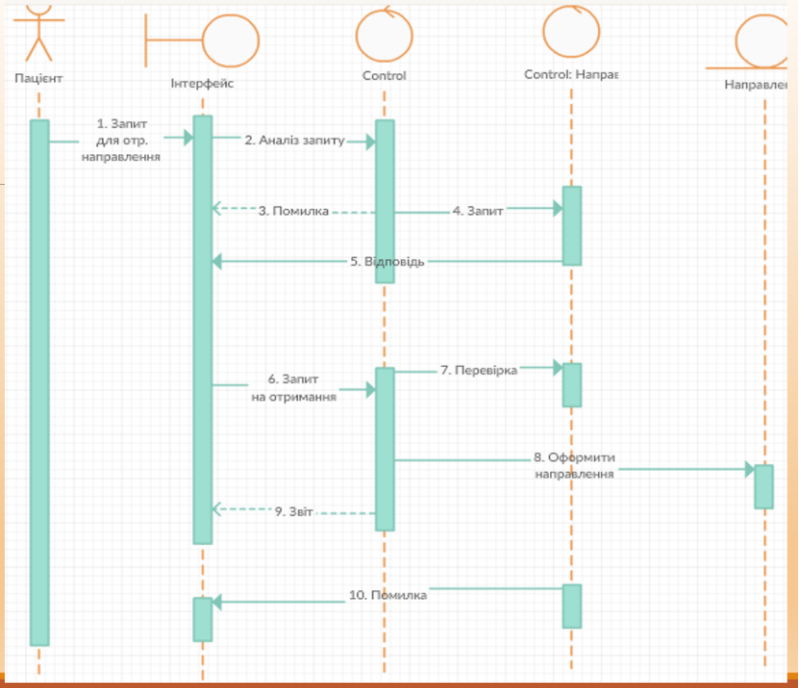
Залежність між модулями пакету Пацієнт

Розробка архітектури програмної системи

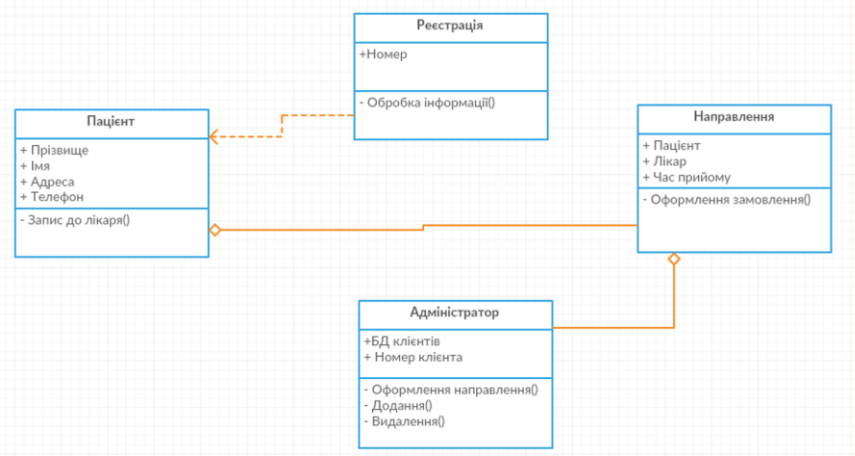


Опис інтерфейсів та сутностей

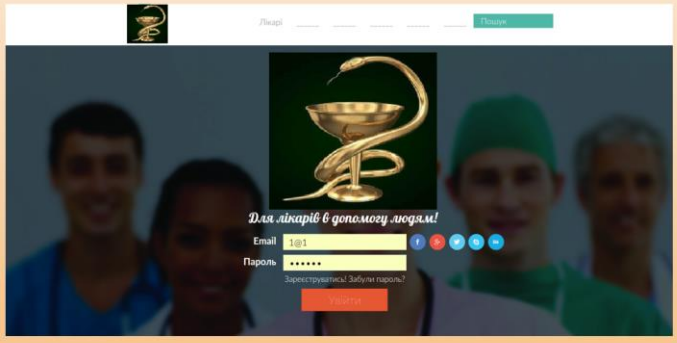
Діаграма послідовності «Оформлення направлення»



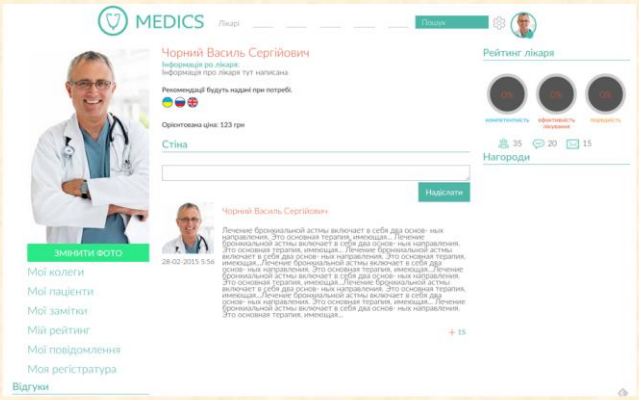
Діаграма класів



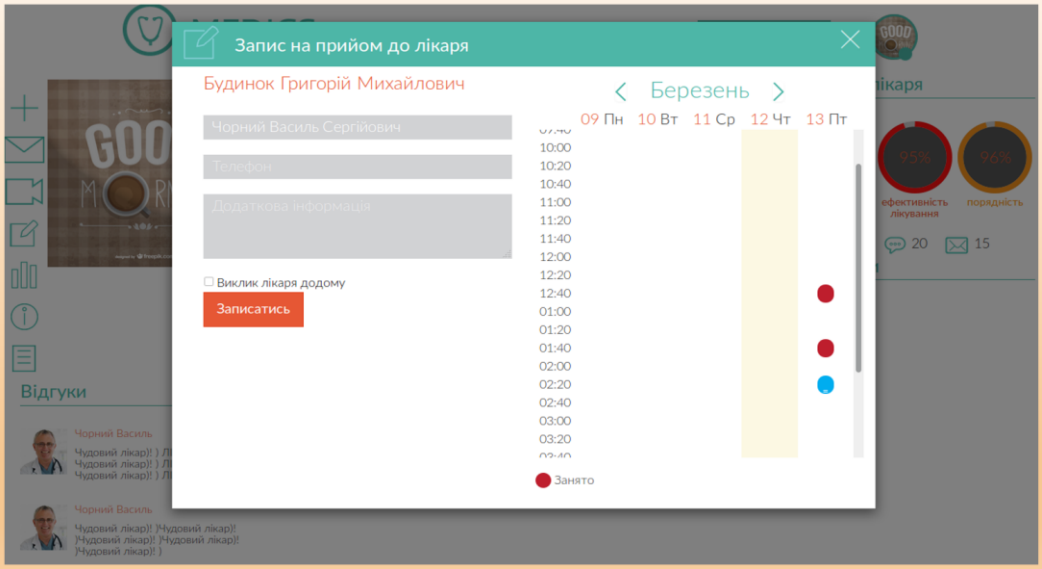
Головна сторінка



Сторінка профілю



Запис на прийом до лікаря



Висновок

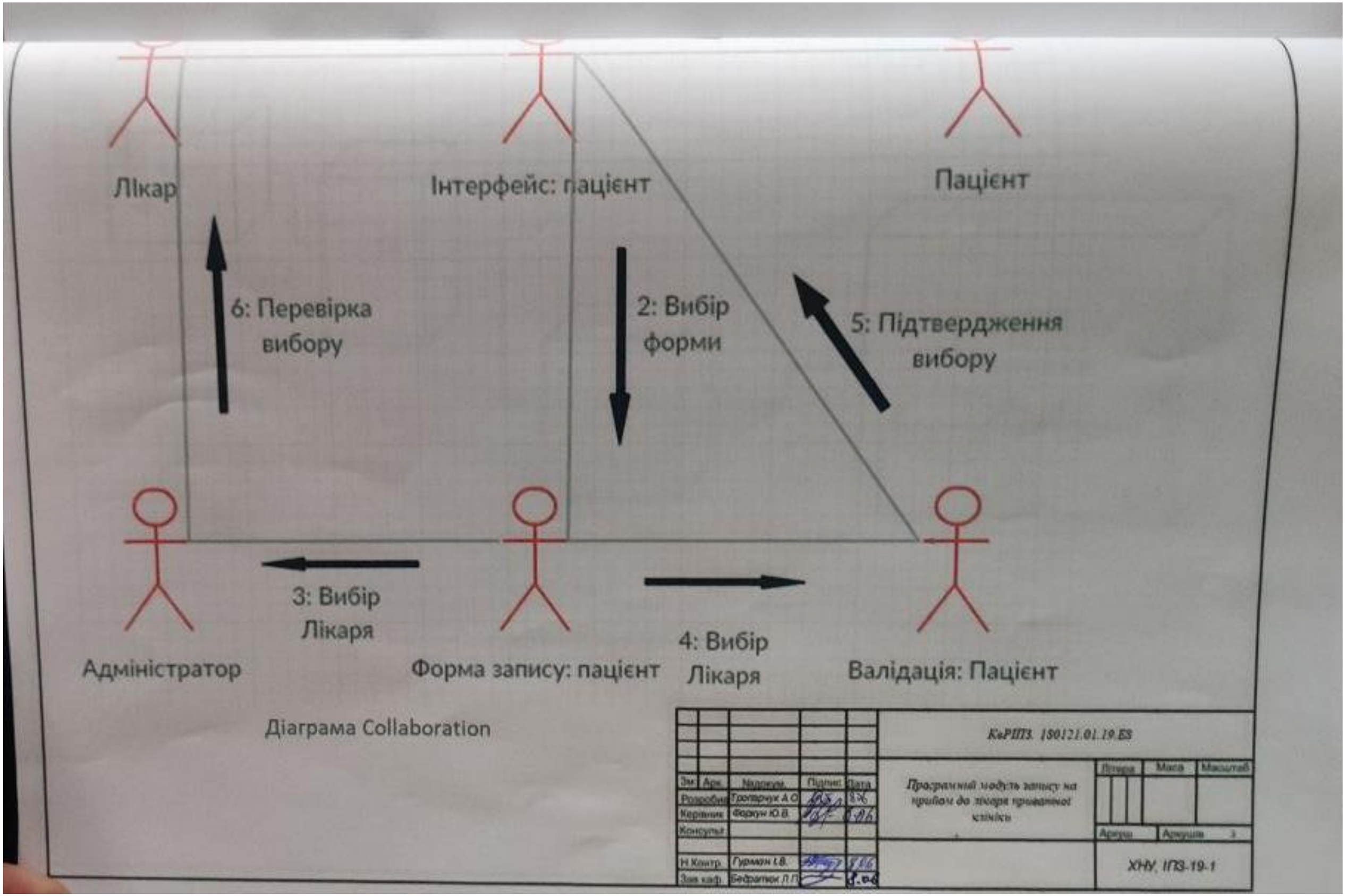
В роботі реалізовано програму автоматизованої системи управління лікувальним закладом. Більшість функцій визначених на етапі проектування вдалося реалізувати у повній мірі.

Автоматизація реалізується за рахунок сучасних засобів введення-виведення інформації, реалізації пошуку та фільтрів.

Переваги проекту:

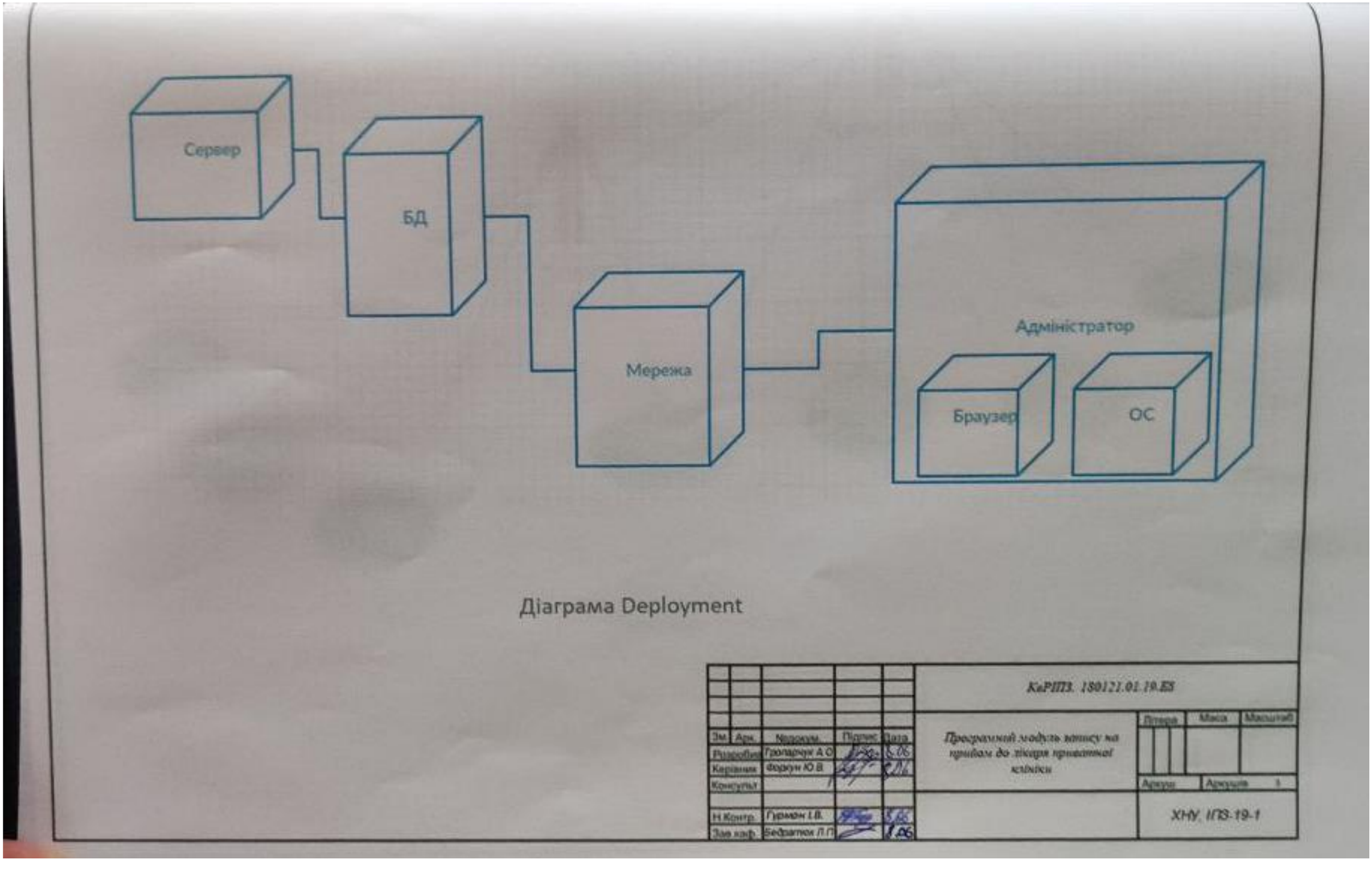
- інтуїтивний інтерфейс користувача;
- зручне додавання, видалення, редагування інформації;
- відсутність прив'язки до певної апаратної платформи;
- відносно низькі системні вимоги.

ГРАФІЧНА ЧАСТИНА



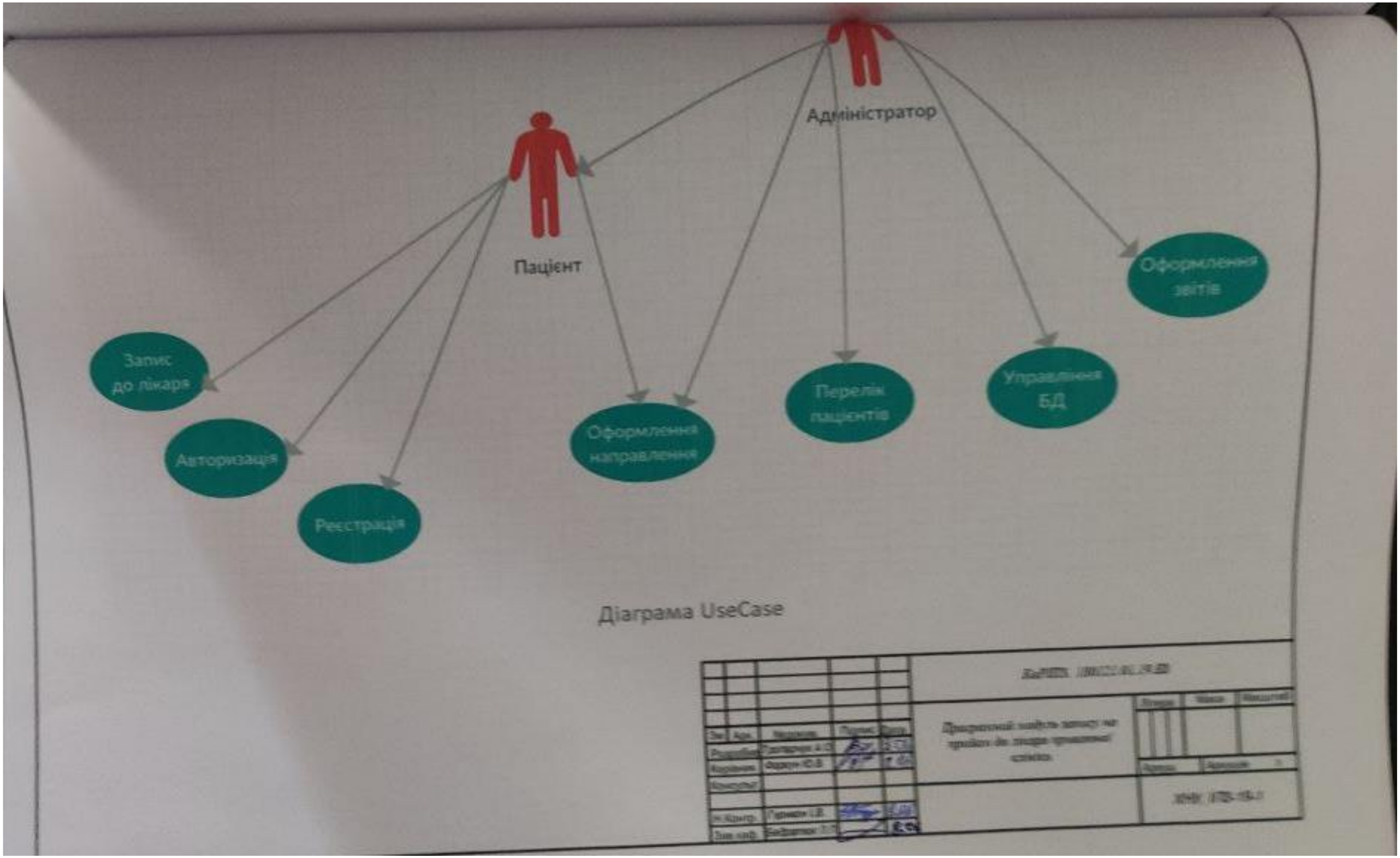
Діаграма Collaboration

					КвРПЗ. 150121.01.19.ПЗ	
Зм.	Арк.	Надпись	Підпис	Дата		
Розробив		Тротярчук А.О.		8.06	Програмний модуль запису на прийом до лікаря приватної клініки	Літера / Маса / Масштаб
Керівник		Фаркун Ю.В.		8.06		Аркуш / Аркушів 1
Консульт.						
Н.Контр.		Гурмэн І.В.		8.06		ХНУ, ІПЗ-19-1
Зав. каф.		Бефратюк Л.П.		8.06		



Діаграма Deployment

					КвРПЗ. 180121.01.19.ЕЗ	
Зм.	Арк.	№докум.	Підпис	Дата	Програмний модуль вводу на прийом до лікаря приватної клініки	Літера / Маса / Масштаб
Розробив		Гропарчук А.О.		5.06		
Керівник		Форкун Ю.В.		5.06		Аркуш / Аркушів 1
Консульт.						
Н.Контр.		Гурман І.В.		5.06		ХНУ, ІПЗ-19-1
Зав.каф.		Бефратюк Л.П.		1.06		



Діаграма UseCase

Завідувачу кафедри
інженерії програмного забезпечення
проф. Бедратюку Л. П.
студента групи ІПЗ-19-1
Тропарчука А.О.
Прізвище, ініціали

ЗАЯВА

Прошу закріпити за мною тему кваліфікаційної роботи освітнього ступеня «бакалавр» за спеціальністю 121 «Інженерія програмного забезпечення»: Програмний модуль запису на прийом до лікаря приватної клініки.

(керівник роботи – Форкун Ю.В.)
Прізвище, ім'я, по батькові

08.06.2023
Дата

Підпис студента

Завідувачу кафедри інженерії програмного забезпечення проф. Бедратюку Л. П.

здобувача вищої освіти

Тропарчук А.О.

Прізвище, ініціали

факультет ІТ, 4 курс, група ІПЗ-19-1

ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності в Хмельницькому національному університеті» від 01.07.2022, згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту і застосування заходів дисциплінарної та академічної відповідальності, ознайомлений. Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений та надаю свою згоду на обробку й збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та/або Anti-Plagiarism) і використання роботи для виявлення академічного плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

08.05

дата

підпис

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ДЕКЛАРАЦІЯ УЧАСНИКА ОСВІТНЬОГО ПРОЦЕСУ

щодо дотримання академічної доброчесності

Цією декларацією я, Прокопчук Андрій Олегович
Прізвище, ім'я, по батькові

студент IVкурсу факультету інформаційних технологій, кафедри інженерії програмного забезпечення
здобувач вищої освіти (шифр та назва спец-ті, курс, академічна група)/ науково-педагогічний працівник (назва кафедри)

назва факультету

підтверджую, що ознайомився (- лась) з Положенням про систему забезпечення академічної доброчесності у Хмельницькому національному університеті та Кодексом академічної доброчесності і **зобов'язуюсь** дотримуватися їх вимог під час освітнього процесу, проведення наукової діяльності, виконання організаційно-адміністративних функцій тощо.

Усвідомлюю, що у разі порушення мною принципів академічної доброчесності нестиму відповідальність перед академічною спільнотою ХНУ згідно з нормами, визначеними Положенням про систему забезпечення академічної доброчесності у Хмельницькому національному університеті, законодавства України

« 05 » лютого 2023 р.

Підпис

Wed Jun 07 14:48:57 EEST 2023, Хіврич Володимир Русланович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 25.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 10%**

	Документ		Сумарний збіг по Базі Даних	
ID: 115087 Назва: БКР Програмний модуль запису на прийом до лікаря приватної клініки Додано в БД: 2023-06-07 Автора: Тропарчук А.О. Керівники: Форкун Ю.В. к.т.н. доц. Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	57807	537	18673 (32%)	188 (35%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми
112153	Назва: Звіт з переддипломної практики бакалавр на тему: Програмний модуль запису на прийом до лікаря приватної клініки Додано в БД: 2023-03-31 Автора: Тропарчук А.О. Керівники: Форкун Ю.В. Консультанти: Опоненти:	14264 (25.0%)	128 (24.0%)

UNICHECK

Ім'я користувача:
Кафедра ІПЗ

ID перевірки:
1015485625

Дата перевірки:
07.06.2023 15:48:00 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
07.06.2023 15:48:46 EEST

ID користувача:
100005589

Назва документа: **Diplom_DrukТропарчук плагіат**

Кількість сторінок: **56** Кількість слів: **9317** Кількість символів: **72919** Розмір файлу: **2.45 MB** ID файлу: **1015142746**

14.4%
Схожість

Найбільша схожість: **8.22%** з джерелом з Бібліотеки (ID файлу: **1015122575**)

7.76% Джерела з Інтернету	908	Сторінка 58
11.2% Джерела з Бібліотеки	92	Сторінка 63

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	5

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ
КАФЕДРИ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатами звіту/звітів подібності щодо роботи, продукованими програмно-технічним засобом (ами) перевірки текстів на плагіат:

Назва: «Програмний модуль запису на прийом до лікаря приватної клініки
Автор: Тропарчук Андрій Олегович
Спеціальність: 121 – Інженерія програмного забезпечення
Освітня програма: Освітньо-професійна програма «Інженерія програмного забезпечення»
Науковий керівник: Форкун Юрій Вікторович, кандидат технічних наук, доцент
Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	**відповідає**
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої й електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того, як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
5	Інше:	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) у тексті кваліфікаційної роботи системами перевірки на плагіат виявлено схожість з деякими документами в частині загальновживаних обов'язкових словосполучень у стандартних бланках (титулка, відомість документів), у структурі змісту, назвах розділів/підрозділів тощо, у назвах публікацій у переліку джерел посилання;

2) в якості запозичень системою було зафіксовано деякі послідовності вихідного коду і посилання на бібліотеки, які є стандартними мовними конструкціями програмування та не можуть розглядатися як об'єкт авторських прав і, відповідно, їх порушення;

3) усі запозичення є фрагментарними або мають належним чином оформлені посилання;

4) виявлені модифікації тексту не впливають на відсоток схожості.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів ідентичності/схожості, складає 14.4% і адресується до титульної сторінки, відомості документів, змісту, заголовків пунктів, літературних джерел та звіту з переддипломної практики, який містить у собі частину кваліфікаційної роботи, що, з урахуванням наведених обґрунтувань, відповідає характеру теми і свідчить на користь кваліфікаційної роботи.

Дата 08.06

Завідувач кафедри Леонід БЕДРАТЮК

Гарант освітньої програми Леонід БЕДРАТЮК

Керівник кваліфікаційної роботи Юрій ФОРКУН

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ
освітнього ступеня «Бакалавр»

Дипломник Тропарчук Андрій Олегович

Тема Програмний модуль запису на прийом до лікаря приватної клініки

Спеціальність 121 – Інженерія програмного забезпечення

Обсяг кваліфікаційної роботи:

Кількість листів креслень 3 ; кількість сторінок записки 80

1.Короткий зміст пояснювальної записки та прийнятих рішень У ході виконання кваліфікаційної роботи була проведено аналіз предметної області, з описом функціональних і нефункціональних вимог. В роботі проаналізовано існуючі альтернативні рішення на ринку, їх переваги та недоліки, що підтверджує актуальність розробки нового програмного забезпечення. В результаті було визначенно різні інструменти для реалізації запропонованих рішень, що дозволило здійснити проектування програмного забезпечення. Результатом роботи є спроектований програмний застосунок, ефективність його роботи доведено тестуванням програмного застосунку, що підтвердило його коректну роботу та готовність до впровадження.

2. Висновок про відповідність роботи поставленому завданню Кваліфікаційна робота виконана відповідно до поставленого завдання та з дотриманням всіх вимог.

3. Характеристика виконання кожного розділу роботи, ступінь використання останніх досягнень науки і техніки та передових методів роботи У вступній частині було показано, що обрана тема є актуальною, визначено мету та завдання дипломного проектування. Перший розділ присвячений детальному аналізу предметної області, в якому було розглянуто наявні рішення та встановлені функціональні та нефункціональні вимоги до розроблюваного програмного забезпечення. У другому розділі проведений аналіз сучасних архітектур, в якому були розглянуті переваги та недоліки, і визначено, що система буде базуватися на монолітній архітектурі та моделі клієнт-сервер. Третій розділ присвячений підготовці всіх залежностей для програмного коду і практичній розробці програмних модулів з описом їх особливостей, що призвело до створення програмного продукту. У цьому розділі проведено функційне тестування системи згідно з функціональними вимогами, що підтвердило правильну роботу застосунку.

4. Позитивні сторони роботи Застосунок має приємний та сучасний дизайн в стилі мінімалізму та в світлих тонах. Елементи інтерфейсу працюють чітко та мають просту та зрозумілу структуру

5. Негативні сторони роботи Необхідність розширити можливість створеного зв'язку з зовнішнім користувачам модуля

6. Оцінка графічного оформлення та пояснювальної записки Графічне оформлення виконано відповідно до теми кваліфікаційної роботи та подано у вигляді діаграм і рисунків. Пояснювальна записка оформлена згідно вимог чинних стандартів.

7. Відгук про кваліфікаційну роботу в цілому Кваліфікаційна робота заслуговує позитивної оцінки, оскільки матеріал пояснювальної записки демонструє чітку структуру, логічну послідовність, зрозумілий та доступний виклад матеріалу. Це дозволяє чітко сприйняти вміст роботи, який гармонійно вписується в контекст тематики роботи. Графічний матеріал, наданий у роботі, надає можливість візуально ознайомитись з деталями проектування системи.

8. Інші зауваження

9. Оцінка кваліфікаційної роботи Кваліфікаційна робота виконана у повному обсязі, відповідає поставленій задачі та заслуговує на оцінку «задовільно».

РЕЦЕНЗЕНТ: к.т.н., доц., доцент кафедри КІІС Нікітюрін Л.О.

"7" 06 2023 р. (підпис)