

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук і технологій

(повне найменування інституту, назва факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ РЕЄСТРАТУРИ ЦЕНТРУ НАДАННЯ
МЕДИЧНОЇ ДОПОМОГИ
SOFTWARE FOR THE MEDICAL RECORDS DEPARTMENT IN A
HEALTHCARE CENTRE

Виконав: студент(ка) 4 курсу, групи КНТ-118сп
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

Агєєв Є.О.

(прізвище та ініціали)

Керівник Скрупський С.Ю.

(прізвище та ініціали)

Рецензент Зеленьова І.Я.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ІРЕ, ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
“ ” 2021 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Агєєва Євгена Олександровича
(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення реєстратури центру надання медичної допомоги

Software for the Medical Records Department in a Healthcare Centre

керівник проєкту (роботи) Скрупський Степан Юрійович, к.т.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “30” березня 2021 року № 103

2. Строк подання студентом проєкту (роботи) 12 травень 2021 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація та тестування програми. 5. Керівництво програміста.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Скрупський С.Ю., доцент		
Нормоконтролер	Андрєєв М.О., асистент		

7. Дата видачі завдання “ 30 ” березня 2021 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз та дослідження предметної області	2 тижні	Розділи 1, 2
3	Розробка архітектури програми	3 тиждень	Розділи 2, 3
4	Розробка програми	4 тижні	Розділ 3
5	Тестування та експериментальне дослідження програми	5 тиждень	Розділ 4
6	Розробка програмної документації	6 тиждень	Розділ 5, додатки
7	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	7 тиждень	Додатки
8	Захист роботи	7 тиждень	

Студент(ка)

_____ Агєєв Є.О.
(підпис) (прізвище та ініціали)

Керівник проєкту (роботи)

_____ Скрупський С.Ю.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 75 с., 39 рис., 3 табл., 3 дод., 14 джерел.

PHP, LARAVEL, MYSQL, WEB-ІНТЕРФЕЙС, ВЕБЗАСТОСУНОК, ЕЛЕКТРОННА РЕЄСТРАТУРА, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, СКБД.

Об'єкт дослідження – процес реєстрації медичних даних.

Предмет дослідження – програмне забезпечення для керування електронними медичними картками.

Метою роботи є розроблення вебзастосунку для керування електронними картками в медичних установах.

Матеріали, методи та технічні засоби: об'єктно-орієнтоване програмування, мова програмування PHP, вебфреймворк Laravel, система управління базою даних MySQL.

Результати. Розроблено програмне забезпечення для підтримки процесу роботи реєстратури центру надання медичної допомоги. В розробленому програмному забезпеченні реалізовано функції реєстрації, автентифікації та авторизації користувачів, CRUD операцій з пацієнтами, доступні тільки лікарям, перегляду персональної інформації, пошуку інформації про пацієнта за його прізвищем, ім'ям, по батькові. Розробленою програмою можуть користуватися працівники реєстратури медичних закладів.

Висновки. Розроблено вебзастосунок для керування електронними картками в медичних установах.

Галузь використання – керування даними пацієнтів медичних установ.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 75 pages, 39 figures, 3 tables, 3 appendixes, 14 sources.

PHP, LARAVEL, MYSQL, WEB INTERFACE, WEB APPLICATION, ELECTRONIC REGISTRATION, SOFTWARE, DBMS.

The object of research is the process of medical data registration.

The subject of research is software for managing electronic medical records.

The aim of the work is to develop a web application for managing electronic cards in medical institutions.

Materials, methods and technical means: object-oriented programming, PHP programming language, Laravel web framework, MySQL database management system.

Results. The software of support the registration process of the medical care center is developed. The developed software implements the functions of registration, authentication and authorization of users, CRUD operations with patients, available only to doctors, viewing personal information, searching for information about the patient by his last name, first name, patronimic. The developed program can be used by employees of the registry of medical institutions

Conclusions. A web application for managing electronic medical records has been developed.

Field of usage is management of patient data of medical institutions.

ЗМІСТ

	С.
Перелік скорочень та умовних познач	8
Вступ	9
1 Аналіз предметної області	11
1.1 Основні принципи документообігу в медичних установах	11
1.2 Огляд існуючих програмних продуктів для автоматизації документообігу в медичних установах	11
1.3 Висновки за розділом 1	15
2 Матеріали і методи	17
2.1 Вибір інструментарію розробки програмного забезпечення	17
2.2 Обґрунтування вибору фреймворку для розробки програмного забезпечення	19
2.3 Обґрунтування вибору середовища розробки програмного забезпечення	20
2.4 Розроблення схеми взаємодії компонентів програмного забезпечення	22
2.5 Висновки за розділом 2	24
3 Опис програми	26
3.1 Вимоги до програми	26
3.2 Алгоритм функціонування розробленої програми	26
3.3 Основні рішення по реалізації компонентів програми	29
3.4 Висновки за розділом 3	31
4 Експлуатація та тестування програми	32
4.1 Призначення програми	32
4.2 Умови виконання програми	32
4.3 Опис роботи користувача з програмним забезпеченням	32
4.4 Повідомлення оператору	36
4.5 Тестування програмного продукту	36
4.6 Висновки за розділом 4	40
5 Керівництво програміста	41

5.1 Призначення й умови застосування програми.....	41
5.2 Характеристики програми	42
5.3 Звернення до програми	42
5.4 Вхідні дані.....	45
5.5 Вихідні дані	45
Висновки.....	46
Перелік джерел посилання	48
Додаток А Технічне завдання.....	50
Додаток Б Текст програми.....	55
Додаток В Слайди презентації	68

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

CRUD	– create, read, update, delete;
CSRF	– cross site request forgery;
IDE	– integrated development environment;
HTML	– hypertext markup language;
HTTP	– hyper text transfer protocol;
ORM	– object-relational mapping;
MVC	– model-view-controller;
REST	– representational state transfer;
ВПЗ	– вільне і відкрите програмне забезпечення;
ЕМК	– електронна медична картка;
ЕОМ	– електронно-обчислювальна машина;
ООП	– об’єктно-орієнтоване програмування;
ПЗ	– програмне забезпечення;
ПО	– предметна область;
СКБД	– система керування базами даних;
ТЗ	– технічне завдання.

ВСТУП

На сьогоднішній день, не дивлячись на активний розвиток інформаційних технологій, більшість медичних закладів ведуть паперовий облік пацієнтів. Після закінчення строку використання, медичні амбулаторні картки відправляються в архів, де визначається строк їх збереження в залежності від їх цінності. Цінність документів визначає експертна комісія, яка може зробити помилку в своїй оцінці. Також, пошук необхідної інформації в архіві займає багато часу, що може стати ключовим фактором у лікуванні людини.

Актуальність створення програми пов'язана з вирішенням проблем паперового документообігу, збереження документів в архівах та швидкого пошуку необхідної інформації.

Мета роботи полягає в розробленні вебзастосунку для керування електронними картками в медичних установах.

Програмний продукт має виконувати такі функції:

- відображення даних;
- редагування даних;
- видалення даних;
- введення даних;
- розподілення ролей користувачів;
- пошук пацієнта;
- захист даних шляхом авторизації користувачів.

Програмний продукт матиме клієнт-серверну архітектуру, яка дозволить користуватися ним на будь-яких ЕОМ з підтримкою веб-браузерів.

Для досягнення мети випускної роботи бакалавра необхідно розв'язати такі завдання:

- проаналізувати предметну область;
- обрати інструментарій розробки ПЗ;
- розробити структурну схему програми;

- розробити ПЗ;
- підготувати програмну документацію на розроблену програму;
- протестувати розроблену програму.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Основні принципи документообігу в медичних установах

Паперовий документообіг в українських лікарнях спричиняє велику незручність як для працівників медичних закладів, так і для їх пацієнтів. Бо витрачається значна кількість часу на фізичне транспортування документів між реєстратурою та кабінетами лікарні, який міг бути використаний на лікування людей [1]-[5].

Одним з основних документів медичної документації є форма первинної облікової документації № 025/о “Медична карта амбулаторного хворого №___”, яка містить всю найбільш використовувану та важливу інформацію про пацієнта [1]-[5].

На даний момент програмних рішень, які б могли вирішити проблему паперового документообігу в українських лікарнях, дуже мало. Усі наявні рішення або розроблені виключно для одного медичного закладу, або вартують значних коштів, бо являть собою апаратно-програмні комплекси.

1.2 Огляд існуючих програмних продуктів для автоматизації документообігу в медичних установах

Широко розповсюдженим рішенням для автоматизації документообігу в медичних установах є продукція компанії «АЛТ Україна Лтд», яка пропонує одразу декілька апаратно-програмних рішень з можливою їх модернізацією. Їх продуктова лінійка складається з:

- ЕМСіMED-Повний – комплексний та повнофункціональний набір інструментів для автоматизації роботи медичного закладу, включає усі доступні модулі та компоненти інформаційної системи;
- ЕМСіЛАБ – лабораторна інформаційна система для автоматизація медичної лабораторії. Найбільше підходить для медичних лабораторій, які не є структурним підрозділом медичного закладу;

- EМСiMED-Поліклініка – автоматизація амбулаторно-поліклінічних закладів охорони здоров'я;
- EМСiMED-Стаціонар – автоматизація робочих процесів лікувального та профілактичного стаціонарів;
- EМСiMED-Light – для організацій на початкових або пілотних стадіях впровадження медичної інформаційної системи. Включає до свого складу лише базові функції. Використовується з метою ознайомлення та початкового навчання персоналу.

Головною особливістю даної компанії є те, що вона пропонує встановлення та налагодження термінальної системи доступу, яка надає підвищену безпеку даних, контроль за роботою користувачів, спрощене адміністрування всієї системи.

Переваги даного апаратно-програмного рішення:

- пропонується одразу цілий комплекс, який здатний задовольнити потреби медичного закладу;
- є можливість підключення окремих модулів для розширення функціональності.

Недоліки:

- складність встановлення;
- велика вартість.

Сервіс «Мій МедКабінет» – є одним з продуктів компанії «АЛТ Україна Лтд» [6]. Даний сервіс частково виконує функції реєстратури медичного закладу, бо він забезпечує тільки запис на прийом до лікаря. Головна сторінка сервісу зображена на рисунку 1.1.

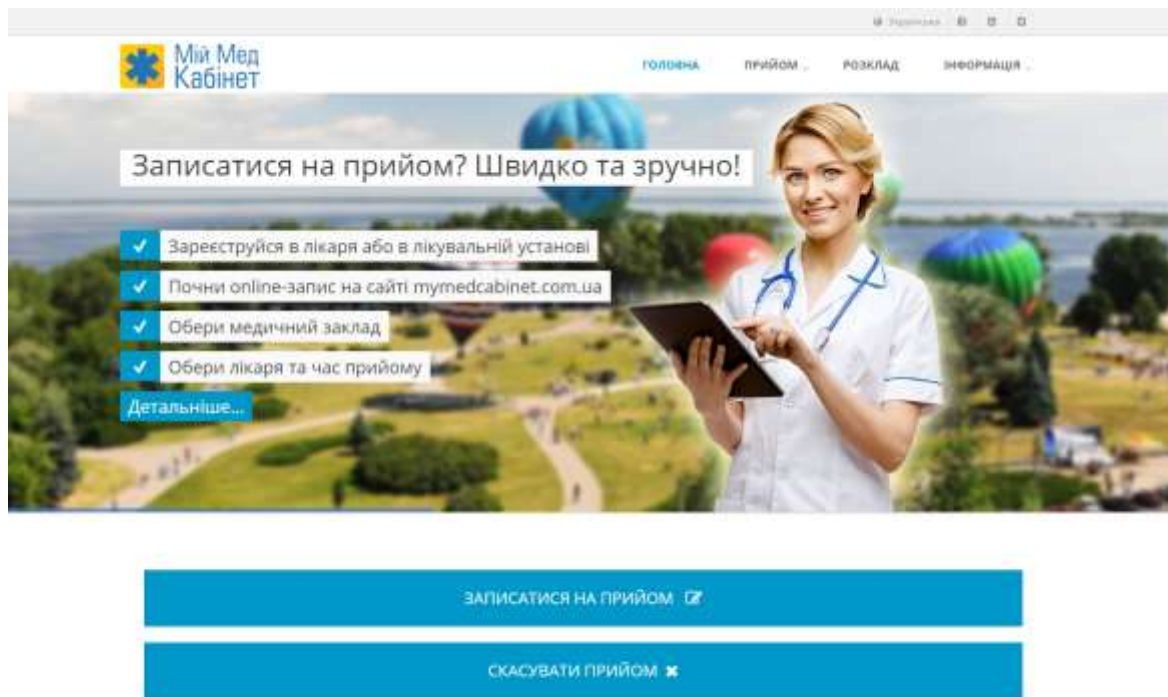


Рисунок 1.1 – Застосунок «Мій МедКабінет»

Переваги даного рішення [6]:

- зручний інтерфейс запису на прийом;
- є частиною апаратно-програмного комплексу компанії «АЛТ Україна Лтд».

Недоліком є відсутність можливості ведення електронної медичної картки.

HELSI.ME – це сучасна, зручна та надійна електронна медична система, створена для пацієнтів, лікарів, державних та приватних медичних закладів [7].

HELSI.ME як і застосунок «Мій МедКабінет» виконує функції реєстрації, але за його допомоги можна не тільки записатися на прийом до лікаря, а й отримати результати аналізів, призначення лікаря та доступ до своєї ЕМК (електронної медичної картки) [7].

Даний застосунок відрізняється від розглянутих вище тим, що він є в певній мірі агрегатором лікарень, бо завдяки одному веб-сайту можна

звернутися за допомогою до різних медичних закладів. Головна сторінка сервісу зображена на рис. 1.2 [7].

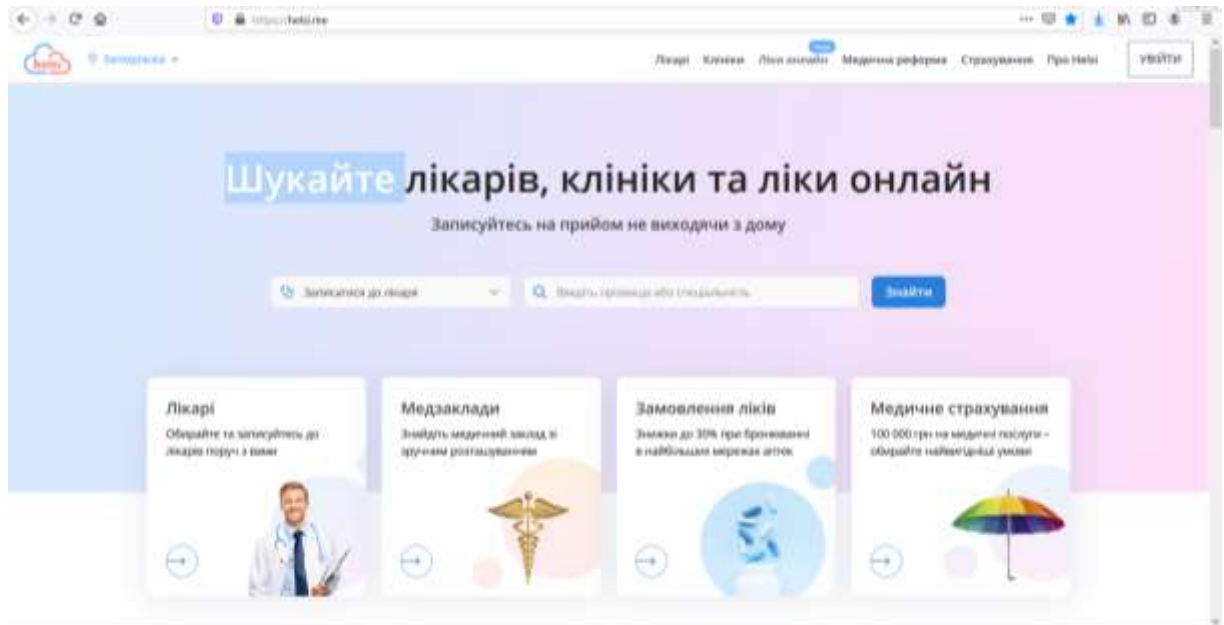


Рисунок 1.2 – Веб-сайт helsi.me

Переваги даного рішення:

- зручний інтерфейс запису на прийом;
- надає уніфіковане рішення.

Недоліком є відсутність можливості встановити окремі модулі для розширення функціональності.

Застосунок health.zp.ua – розроблений для департаменту охорони здоров'я запорізької міської ради і є не комерційним продуктом [8]. Даний застосунок виконує роль агрегатора лікарень міста Запоріжжя, який служить для пошуку медичних закладів у місті, запису на прийом до лікарів та виклику невідкладної медичної допомоги. Також health.zp.ua містить довідкову інформацію про медичні програми та вакцинації [8].

До основних недоліків даного сервісу можна віднести відсутність системи електронного документообігу та непрацюючі посилання. Головна сторінка сервісу зображена на рис. 1.3 [8].

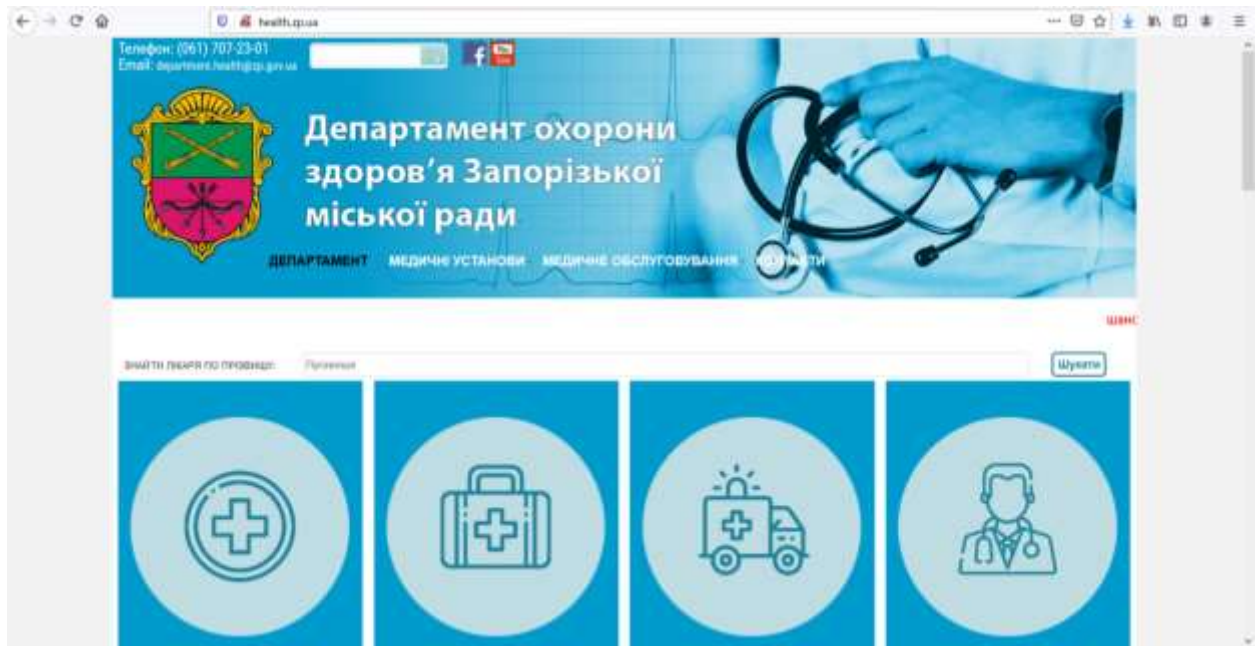


Рисунок 1.3 – Веб-сайт health.zp.ua

Переваги даного рішення [8]:

- зручний інтерфейс запису на прийом;
- надає вичерпну інформацію про медично-лікувальні заклади Запоріжжя.

Недоліки:

- некомерційне рішення;
- немає можливості ведення електронної медичної картки;
- наявні помилки при виконанні.

1.3 Висновки за розділом 1

Виконано огляд проблеми автоматизації процесу документообігу в медичних установах. Проаналізовано програмне забезпечення для автоматизації документообігу. Виявлено, що компанія «АЛТ Україна Лтд» пропонує комплексне апаратно-програмне рішення, яке може повністю задовольнити потреби. Проте його впровадження та навчання персоналу роботи з ним займе значну кількість часу, і потребуватиме вагомих фінансових затрат.

Застосунок HELSI.ME пропонує тільки програмне рішення, яке найбільш помітно, для пацієнтів, покращать якість обслуговування. Застосунок health.zp.ua нажаль як і більшість державних веб-сайтів та сервісів страждає від низької продуктивності та якості роботи. Також, даний застосунок не пропонує жодного вирішення проблеми паперової документації. Хоча, до переваг даного сервісу можна віднести зручний запис на прийом до лікаря.

Тому доцільно створити вебзастосунок, який буде орієнтований на зменшення паперового документообігу в реєстратурі медичного закладу, усунення недоліків існуючих програмних рішень та автоматизації роботи лікаря шляхом створення ЕМК.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Вибір інструментарію розробки програмного забезпечення

Сьогоднішній світ інформаційних технологій значною мірою орієнтований на різні веб-сервіси та взаємодію Інтернет мережею. В основі більшості веб-сервісів лежить MVC(model-view-controller) шаблон, яка передбачає у своїй структурі [9]:

- модель – це центральний компонент шаблону MVC і відображає поведінку застосунку, незалежну від інтерфейсу користувача. Модель стосується прямого керування даними, логікою та правилами застосунку [9];
- вигляд – може являти собою будь-яке представлення інформації, одержуване на виході, наприклад графік чи діаграму. Одночасно можуть співіснувати кілька виглядів (представлень) однієї і тієї ж інформації [9];
- контролер – одержує вхідні дані й перетворює їх на команди для моделі чи вигляду [9].

PHP, попередня назва: Personal Home Page Tools — скриптова мова програмування, була створена Расмусом Лерддорфом для генерації HTML-сторінок на стороні веб-сервера. PHP є однією з найпоширеніших мов, що використовуються у сфері веб-розробок (разом із Java, .NET, Python, Ruby) [10]-[12].

PHP інтерпретується веб-сервером у HTML-код, який передається на сторону клієнта. На відміну від скриптової мови JavaScript, користувач не бачить PHP-коду, бо браузер отримує готовий html-код. Це є перевага з точки зору безпеки, але погіршує інтерактивність сторінок. Але PHP дає можливість генерувати JavaScript-коди, які виконуються вже на стороні клієнта [10]-[12].

Популярність в області побудови веб-сайтів визначається наявністю великого набору вбудованих засобів для розробки веб-додатків [10]-[12]. Основні з них [10]-[12]:

- автоматичне вилучення POST і GET-параметрів, а також змінних оточення веб-сервера в визначені масиви;
- взаємодія з великою кількістю різних СКБД (MySQL, MySQLi, SQLite, PostgreSQL, Oracle (OCI8), Oracle, Microsoft SQL Server, Sybase, mSQL, IBM DB2, Cloudscape і Apache Derby, Informix, Ovrimos SQL, Lotus Notes , DB ++, DBM, dBase, DBX, FrontBase, FilePro, Ingres II, SESAM, Firebird / InterBase, Paradox File Access, MaxDB), Redis [10]-[12];
- автоматизована відправка HTTP-заголовків;
- робота з HTTP-авторизацією;
- робота з cookies та сесіями;
- робота з локальними і віддаленими файлами, сокетом;
- обробка файлів, що завантажуються на сервер.

З поміж всього можна виділити те, що для PHP створено безліч фреймворків, які можуть значно прискорити розробку проекту [10]-[12].

Порівняльна характеристика мов програмування наведена у таблиці 2.1.

Таблиця 2.1 – Порівняння мов програмування

Критерій порівняння	PHP	JavaScript	Java
Можливість додати в HTML	+	+	+
Підтримка ООП	+	±	+
Підтримка прототипів	–	+	–
Підтримка процедурного програмування	+	–	–
Статична типізація	–	–	+
Динамічна типізація	+	+	–
Одночасне виконання паралельних задач	+	–	+

Виходячи з приведеного порівняння у таблиці представлених вище, ми можемо стверджувати, що сильну перевагу в створенні веб-додатку на обрану тему має PHP завдяки гарній підтримці ООП та динамічній типізації, яка значно може скоротити час розробки застосунку.

2.2 Обґрунтування вибору фреймворку для розробки програмного забезпечення

Для розробки програмного забезпечення доцільно обрати фреймворк Laravel, який є безкоштовним, з відкритим кодом PHP-фреймворком, призначеним для розробки вебзастосунків відповідно до шаблону MVC (model–view–controller). Деякі з особливостей Laravel є модульна система упакування з виділеним менеджером залежностей, різні способи для доступу до реляційних баз даних, утиліти, які допомагають в розгортанні додатків і технічного обслуговування [13].

Ключові особливості, що лежать в основі архітектури Laravel [10]-[13]:

- пакети (англ. Packages) – дозволяють створювати і підключати модулі в форматі Composer до додатка на Laravel. Багато додаткових можливостей вже доступно у вигляді таких модулів [13];

- Eloquent ORM – реалізація шаблону проектування ActiveRecord на PHP. Дозволяє строго визначити відносини між об'єктами бази даних. Стандартний для Laravel будівник запитів Fluent підтримується ядром Eloquent [13];

- логіка програми – частина розроблюваного застосунку, оголошена або за допомогою контролерів, або маршрутів (функцій-замикань). Синтаксис оголошень схожий на синтаксис, який використовується в каркасі Sinatra [13];

- зворотна маршрутизація пов'язує між собою згенеровані додатком посилання і маршрути, дозволяючи змінювати останні з

автоматичним оновленням пов'язаних посилань. При створенні посилань за допомогою іменованих маршрутів Laravel автоматично генерує кінцеві URL;

- REST-контролери – додатковий шар для поділу логіки обробки GET- і POST-запитів HTTP;

- автозавантаження класів – механізм автоматичного завантаження класів PHP без необхідності підключати файли їх визначень в include. Завантаження на вимогу запобігає завантаження непотрібних компонентів; завантажуються тільки ті з них, які дійсно використовуються [13];

- укладачі уявлень (англ. View composers) – блоки коду, які виконуються при генерації уявлення (шаблону);

- інверсія управління (англ. Inversion of Control) – дозволяє отримувати екземпляри об'єктів за принципом зворотного управління. Також може використовуватися для створення і отримання об'єктів-одинаків;

- міграції – система управління версіями для баз даних. Дозволяє зв'язувати зміни в коді програми зі змінами, які потрібно внести в структуру БД, що спрощує розгортання і оновлення програми;

- модульне тестування – грає дуже велику роль в Laravel, який сам по собі містить велику кількість тестів для запобігання регресій (помилкам внаслідок поновлення коду або виправлення інших помилок);

- сторінковий висновок (англ. Pagination) – спрощує генерацію сторінок, замінюючи різні способи вирішення цього завдання єдиним механізмом, вбудованим в Laravel [13].

2.3 Обґрунтування вибору середовища розробки програмного забезпечення

Для розробки вебзастосунків доцільно обрати PHPStorm, що представляє собою IDE (Integrated Development Environment) для створення вебзастосунків за допомогою мови програмування PHP, яка де-факто стала індустріальним стандартом. Це середовище розробки створено компанією

JetBrains на основі платформи IntelliJ IDEA, написаної на Java. Користувачі можуть розширити функціональність середовища розробки за рахунок установки плагінів, розроблених для платформи IntelliJ, або написавши власні плагіни [14].

PhpStorm являє собою інтелектуальний редактор для PHP, HTML і JavaScript з можливостями аналізу коду на льоту, запобігання помилок у сирцевому коді і автоматизованими засобами рефакторинга для PHP і JavaScript. Автодоповнення коду в PhpStorm підтримує специфікацію PHP 5.3/5.4/5.5/5.6/7.0/7.1, включаючи генератори, співпрограми, простори імен, замикання, типажі і синтаксис коротких масивів. Присутній повноцінний SQL-редактор з можливістю редагування отриманих результатів запитів [14].

Середовище розробки включає [14]:

- підтримку SQL і баз даних (рефакторинг схеми бази даних, генерація скриптів міграції схеми, експорт результатів виконання запиту у файл або буфер обміну, редагування збережених процедур і багато іншого) [14];
- віддалене розгортання додатків і автоматичну синхронізацію з використанням FTP , SFTP , FTPS та інших протоколів [14];
- інтеграцію з системами управління версіями (Git, Subversion, Mercurial, Perforce, CVS, TFS), що дозволяє робити дій такі, як commit, merge, diff та інші, прямо з PhpStorm [14];
- локальну історію (локально відстежує будь-які зміни в коді);
- PHP UML (діаграми класів UML для PHP коду з рефакторингом, що викликаються прямо з діаграми) [14];
- підтримку Phing (автодоповнення, перевірка стандартних тегів, властивостей, імен цілей, значень атрибутів шляху в компоновальних файлах);
- інтеграцію з системами відстеження помилок;
- підтримку Vagrant, SSH консолі і віддалених інструментів;
- підтримку Google App Engine For PHP;

– PhpStorm також дозволяє різні поєднання клавіш для підвищення ефективності.

До особливостей PhpStorm можна віднести можливість простого запуску модульних тестів, гарну підтримку CSS, JavaScript і HTML, на відміну від Eclipse та NetBeans. Також, оскільки PhpStorm є комерційним продуктом, він має грану підтримку та отримує регулярні оновлення [14].

Порівняльна характеристика середовищ розробки наведена у таблиці 2.2.

Таблиця 2.2 – Порівняння середовищ розробки

Критерій порівняння	Eclipse	PHPStorm	NetBeans
Безкоштовність	+	±	+
Інтелектуальний рефакторинг	–	+	±
Розуміння контексту(autocomplete)	–	+	±
Пошук використання(Find Usage)	–	+	–
Кросплатформність	+	+	+
Вбудований дебагер	–	+	–

Як видно з порівняльної таблиці, PhpStorm має найширші функціональні можливості з поміж своїх конкурентів.

2.4 Розроблення схеми взаємодії компонентів програмного забезпечення

Вебзастосунки найбільш доцільно розроблювати з використанням архітектурного шаблону MVC. Завдяки цьому отримується модульність програми з високою стійкістю до змін та масштабованість проекту в цілому. Архітектурний шаблон MVC передбачає розподіл вебзастосунку на три складові.

Першою складовою є Model (модель), яка представляє собою центральний компонент шаблону MVC. Ця складова відображає поведінку вебзастосунку, яка є незалежною від інтерфейсу користувача. Компонент «Модель» відноситься до прямого керування даними, а також логікою та правилами вебзастосунку.

Компонент View (представлення) призначений для представлення інформації, що буде одержуватися на виході, наприклад у вигляді графіків або діаграм. В один і той самий час можуть співіснувати декілька видів (представлень) однієї і тієї ж самої інформації, наприклад, такими представленнями можуть бути гістограма для керівництва компанії та таблиці для бухгалтерії.

Складова моделі MVC Controller (контролер) призначена для забезпечення зв'язку між користувачем та розроблюваною програмою. Ця складова контролює та направляє дані від користувача програмного забезпечення до ПЗ і навпаки.

На рис. 2.1 зображено схему взаємодії компонентів програмного забезпечення.

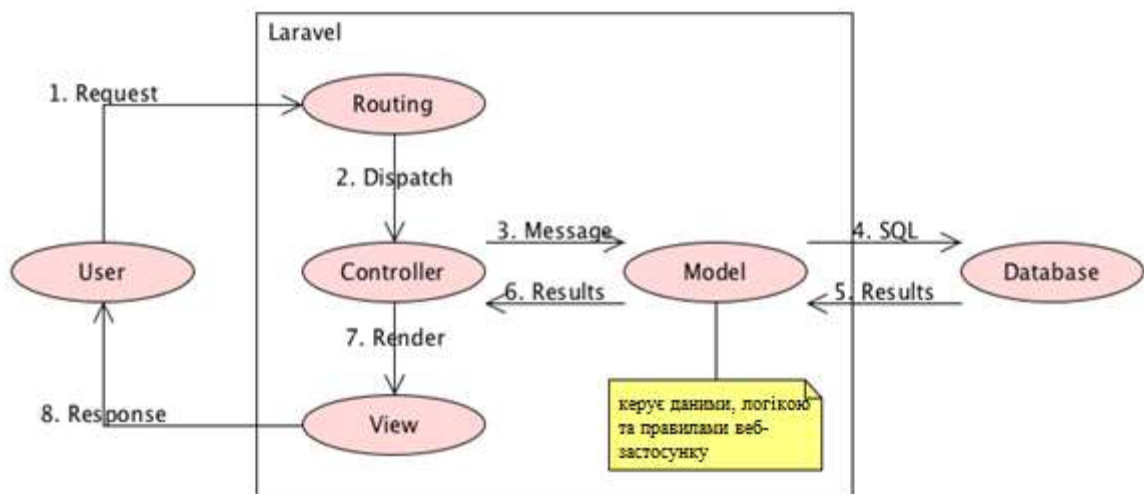


Рисунок 2.1 – Схема взаємодії компонентів програмного забезпечення

Request (запит) – представляє собою повідомлення, що надсилає веббраузер клієнта до сервера відповідно HTTP протоколу.

Routing (маршрутизація) – призначений для отримання запиту від вебклієнту, виконує відправку запиту клієнта до відповідного контролера для подальшого оброблення цього запиту. Цей компонент може виконувати авторизацію для можливості виконання певних дій.

Controller (контролер) – компонент, який містить основну логіку щодо відповідей на запит, отриманий від клієнта. Компонент Controller формує відповідь та представлення для клієнта. При необхідності компонент Controller взаємодіє з моделлю.

Model (модель) – компонент, який представляє собою Eloquent ORM, що створена Laravel та призначена для забезпечення можливості реалізації шаблону Active Record, який використовується для взаємодії з БД. Кожен клас шаблону Active Record відображає таблицю БД та забезпечує реалізацію загальних CRUD операцій [13].

View (представлення) – використовує шаблонізатор Blade для формування представлення на основі даних, які подає компонент контролер.

Response (відповідь) представляє собою відповідь вебсервера на запит вебклієнта згідно HTTP протоколу, що подається у вигляді HTML сторінки.

2.5 Висновки за розділом 2

Обґрунтовано вибір мови програмування PHP для розробки програмного забезпечення. Проаналізовано її переваги та недоліки, проведено порівняльний аналіз з мовами JavaScript та Java. Мову PHP обрано, враховуючи її переваги щодо зручності розробки, об'єктно-орієнтовних можливостей та часу створення вебзастосунків.

Для розробки програмного забезпечення обрано фреймворк Laravel, який є безкоштовним, з відкритим кодом PHP-фреймворком, призначеним для розробки вебзастосунків відповідно до шаблону MVC.

Проведено аналіз середовищ розробки Eclipse, PHPStorm, NetBeans. У результаті якого було визначено, що PHPStorm найкраще підходить для створення вебзастосунків за допомогою мови програмування PHP.

Визначено загальну структуру програми та побудовано архітектуру системи, яка відповідає меті роботи, та може реалізувати обрані функції за допомогою раніше визначених інструментів (програмних продуктів). Розроблено структурну схему програми у вигляді набору компонентів та зв'язків між ними.

3 ОПИС ПРОГРАМИ

3.1 Вимоги до програми

Розроблювана програма повинна бути вебзастосунком, призначеним для роботи у медичних закладах, тому має підвищені вимоги щодо забезпечення безпеки даних користувачів та пацієнтів. Для варто реалізувати захист від основних видів атак та вразливостей – CSRF (cross site request forgery), XSS (cross-site scripting), SQL (structured query language) injection та інших.

Програма повинна забезпечувати наступні вимоги:

- коректна передача та прийом даних за допомогою веб-інтерфейсу;
- програма повинна мати «дружній» для користувача інтерфейс;
- розділення на дві ролі: лікар та пацієнт;
- лікарю доступні CRUD операції з будь-якою інформацією про пацієнта;
- пацієнту тільки перегляд персональної інформації;
- має бути можливість швидкого пошуку пацієнтів за ПІБ;
- розміри елементів управління повинні змінюватися відповідно до зміни розмірів робочого вікна;
- час обробки зображення не повинно перевищувати 10 секунд;
- час реакції програми на натискання будь-якої з клавіш і маніпуляції мишею не повинно перевищувати 1 секунди, в разі відповідності системних ресурсів вимогу до складу і параметрів технічних засобів.

3.2 Алгоритм функціонування розробленої програми

Основні алгоритми, виконання яких можливе при роботі програми, зображені на рис. 3.1.

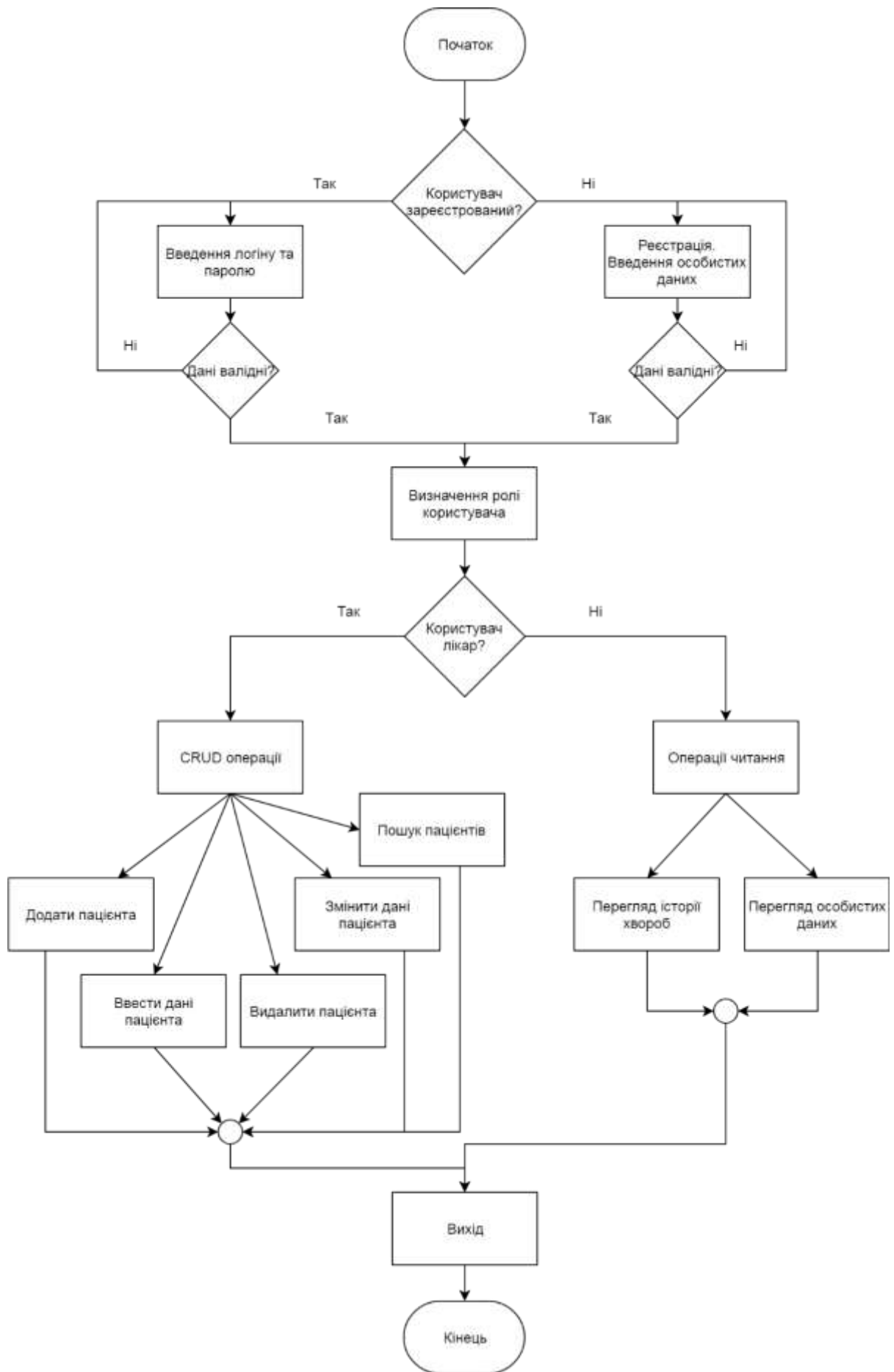


Рисунок 3.1 – Функціональна схема програми

Як видно зі схеми зображеної вище, під час розробки були використані такі алгоритми:

- алгоритм авторизації;
- алгоритм автентифікації;
- алгоритм реєстрації;
- алгоритм валідації;
- алгоритм CRUD операцій;
- алгоритм пошуку пацієнтів;
- алгоритм перегляду даних.

На початку роботи з вебзастосунком користувач потрапляє на головну сторінку, де надається основна інформація про застосунок та пропонується або автентифікуватися, або зареєструватися.

Для того щоб зареєструватися необхідно ввести логін, електронну адресу, пароль та підтвердження паролю. Електронна адреса має бути унікальною, тобто не може існувати декілька користувачів з однаковою адресою. Пароль повинен складатися щонайменше з шести символів.

Для автентифікації необхідно ввести свою електронну адресу та пароль. У разі необхідності можна скинути пароль.

Для реалізації реєстрації, автентифікації та скидання паролю було використано попередньо згенеровані фреймворком Laravel контролери.

Авторизація користувачів здійснюється за допомогою вбудованого в Laravel механізму авторизації – Gate(ворота). Завдяки цьому механізму визначається, які дії дозволені певній групі користувачів. Групи користувачів заздалегідь визначені у СКБД MySQL.

Перевірка введених даних відбувається за допомогою двоетапної валідації. Спочатку перевірка проводиться на стороні клієнта засобами HTML, але вона не є надійною. Тому відбувається ще одна перевірка на стороні сервера за допомогою засобів Laravel, а саме класу `ValidatesRequests`.

Для лікарів доступні CRUD функції для додавання, зміни, перегляду та видалення інформації про пацієнтів, інфекційні захворювання,

алергологічний анамнез, щоденник захворювань, щорічний епікриз на диспансерних хворих, непереносимість до лікарських препаратів, заключні діагнози, листки тимчасової непрацездатності, госпіталізації, хірургічних втручання, вакцинацію.

Пацієнтам доступний тільки перегляд інформації, яка стосується тільки їх.

Пошук пацієнтів реалізований за допомогою механізму Full Text Search (повнотекстовий пошук) у СКБД MySQL. Даний механізм дозволяє робити вибірку з БД з пошуком ключових слів одразу в декількох полях.

В таблиці пацієнтів на основі трьох полів (прізвище, ім'я, по батькові) створено індекс типу FULLTEXT, який і служить для пошуку пацієнтів.

3.3 Основні рішення по реалізації компонентів програми

Структурна схема компонентів програми може бути представлена у вигляді зображеному на рис. 3.2.

У каталозі App міститься основний код програми. Майже всі класи програми розміщено у цьому каталозі.

Компонент Console містить всі спеціальні команди Artisan. Ці команди можуть бути створені за допомогою команди `command: make`. У цьому каталозі також знаходиться консольне ядро застосунку, де зареєстровані спеціальні команди Artisan та визначені заплановані завдання.

У компоненті Exceptions міститься обробник винятків програми, і це також хороше місце для зберігання будь-яких винятків, викинутих програмою. Для зміни налаштувань того, як зберігаються або оброблюються винятки необхідно змінити клас Handler в цьому каталозі.

Компоненті HTTP містить контролери, middleware (проміжне програмне забезпечення). Майже вся логіка для обробки запитів, що надходять до застосунку, розміщена в цьому компоненті.

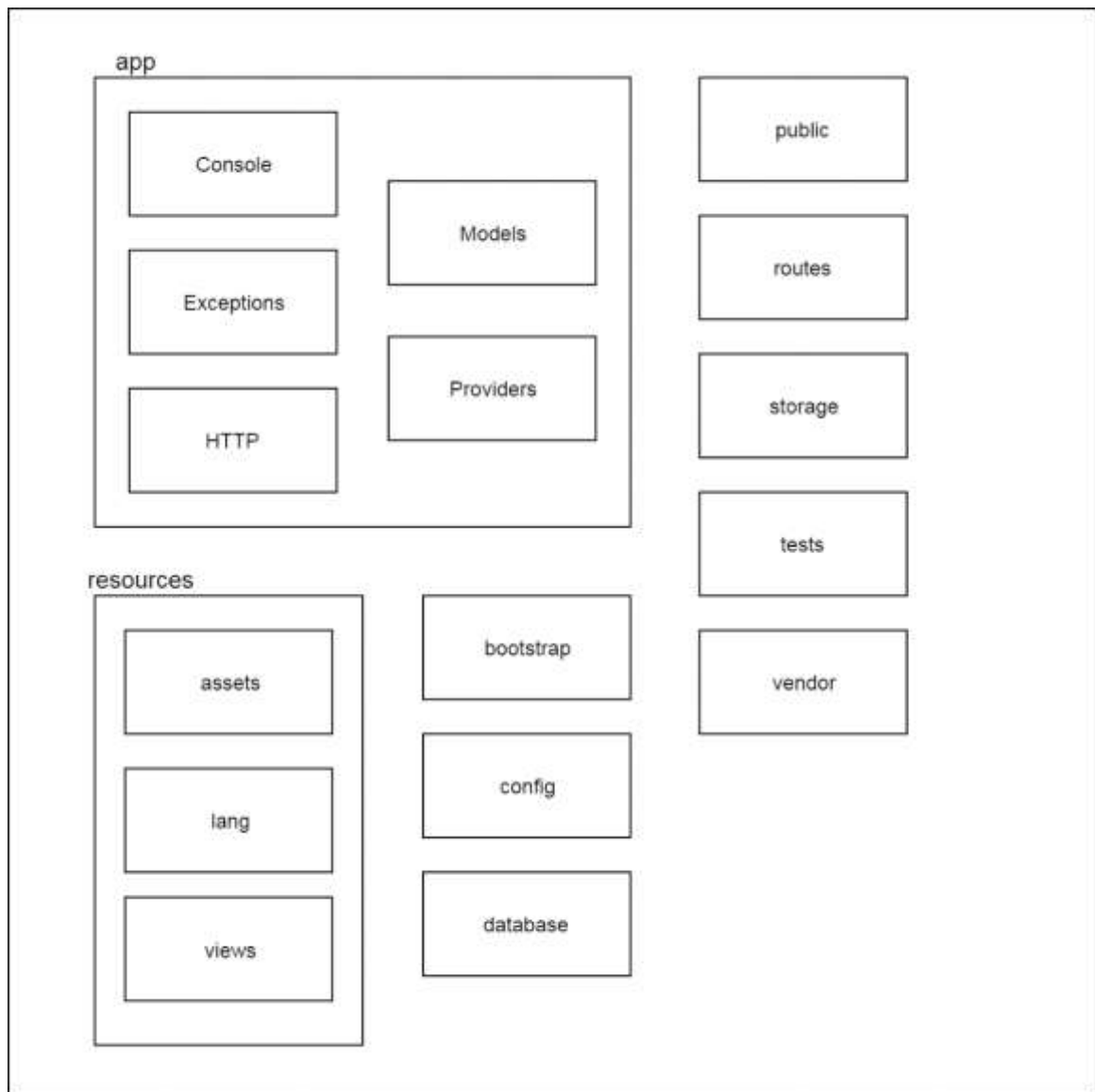


Рисунок 3.2 – Структурна схема компонентів програми

Компонент **Models** містить усі наявні у застосунку моделі відповідальні за взаємодію з БД.

Компонент **Providers** містить усіх «постачальників послуг» для розробленої програми. Постачальники послуг завантажують в програму, шляхом зв'язування служб у контейнері служб, реєстрування подій або виконуючи будь-які інші дії для підготовки застосунку до вхідних запитів.

У каталозі **Resources** містяться всі представлення, а також сирі, некомпільовані ресурси, такі як LESS, SASS або JavaScript. Також містить всі файли необхідні для мультимовної системи.

Компонент Config, як видно з його назви, містить файли конфігурації програми.

Компонент Storage містить скомпільовані шаблони Blade, сесії на основі файлів, кеш-файли та інші файли, створені фреймворком. Цей компонент розділений на директорії: app, framework та logs. Каталог app може бути використаний для зберігання будь-яких файлів, створених програмою. Каталог framework використовується для зберігання згенерованих файлів і кешу. А logs логів містить файли журналів програми.

Компонент Vendor містить залежності Composer.

3.4 Висновки за розділом 3

Проаналізовано основні вимоги до програмного забезпечення реєстратури центру надання медичної допомоги. Наведено функціональну схему програмного забезпечення реєстратури центру надання медичної допомоги.

Описані основні компоненти побудованого програмного забезпечення за допомогою мови програмування PHP та фреймворку Laravel.

4 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМИ

4.1 Призначення програми

Програмне забезпечення призначене для ведення та зберігання ЕМК в медичних установах. Метою створення програми було зменшити паперовий документообіг в медичних установах та покращити якість діагностування пацієнтів, шляхом впровадження швидкого доступу до всієї необхідної інформації про хворого. Це забезпечується наданням засобів для внесення відповідних даних про пацієнтів медичних установ, їх перегляду, а також й наданням засобів пошуку та інших засобів.

4.2 Умови виконання програми

Технічні засоби, за допомогою яких може експлуатуватися програма, повинні включати такі, що наведені нижче. У клієнта повинна бути наявність на пристрої доступу до Інтернет мережі та сучасного веб-браузера.

До складу технічних засобів повинен входити сервер, що задовольняє наступним вимогам:

- процесор з тактовою частотою не менше 2 GHz;
- оперативна пам'ять об'ємом 2 Gb, або більше;
- накопичувач об'ємом не менше 120 Gb;
- підтримка сервером MySQL версії не нижче 5.0;
- підтримка Apache HTTP Server, або nginx, або аналогічного веб-серверу;
- підтримка PHP версії не нижче 7.2.

4.3 Опис роботи користувача з програмним забезпеченням

Для запуску застосунку на локальному сервері з ОС Windows та встановленою збіркою веб-сервера XAMPP необхідно виконати наступні дії.

Створити віртуальний хост. Для цього відкривається конфігураційний файл Apache, який називається `httpd-vhosts.conf`, де визначається розташування проекту з вебзастосунком та назва, за якою його можна знайти в веб-браузері (рис. 4.1).

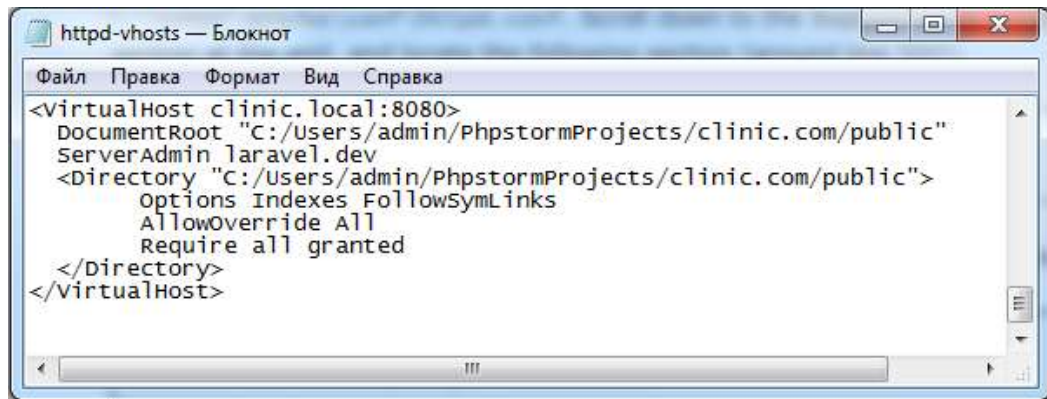


Рисунок 4.1 – Конфігураційний файл `httpd-vhosts.conf`

Після цього необхідно зарезервувати ім'я вебзастосунку на локальній машині. Для цього в файлі `C:\WINDOWS\system32\drivers\etc\hosts` вписується локальна адреса та резервоване ім'я (рис. 4.2).

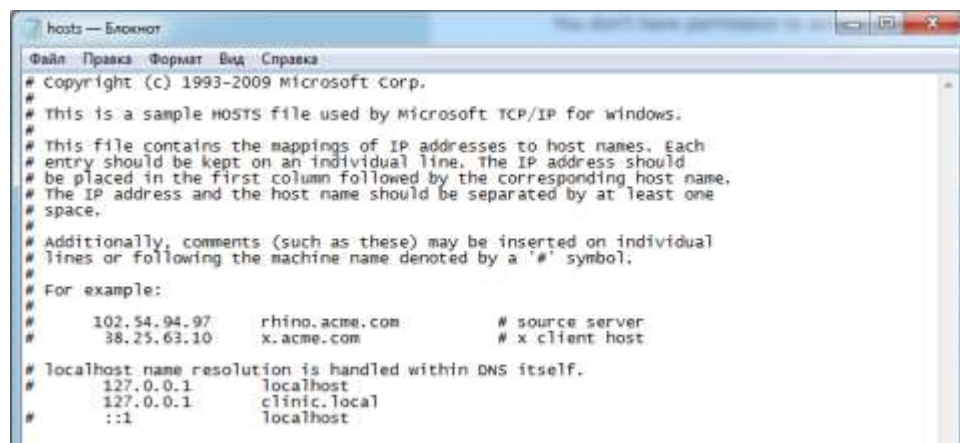


Рисунок 4.2 – Налаштування файлу `hosts`

Далі необхідно відкрити проект в PHPStorm та вказати з'єднання до БД (рис. 4.3).

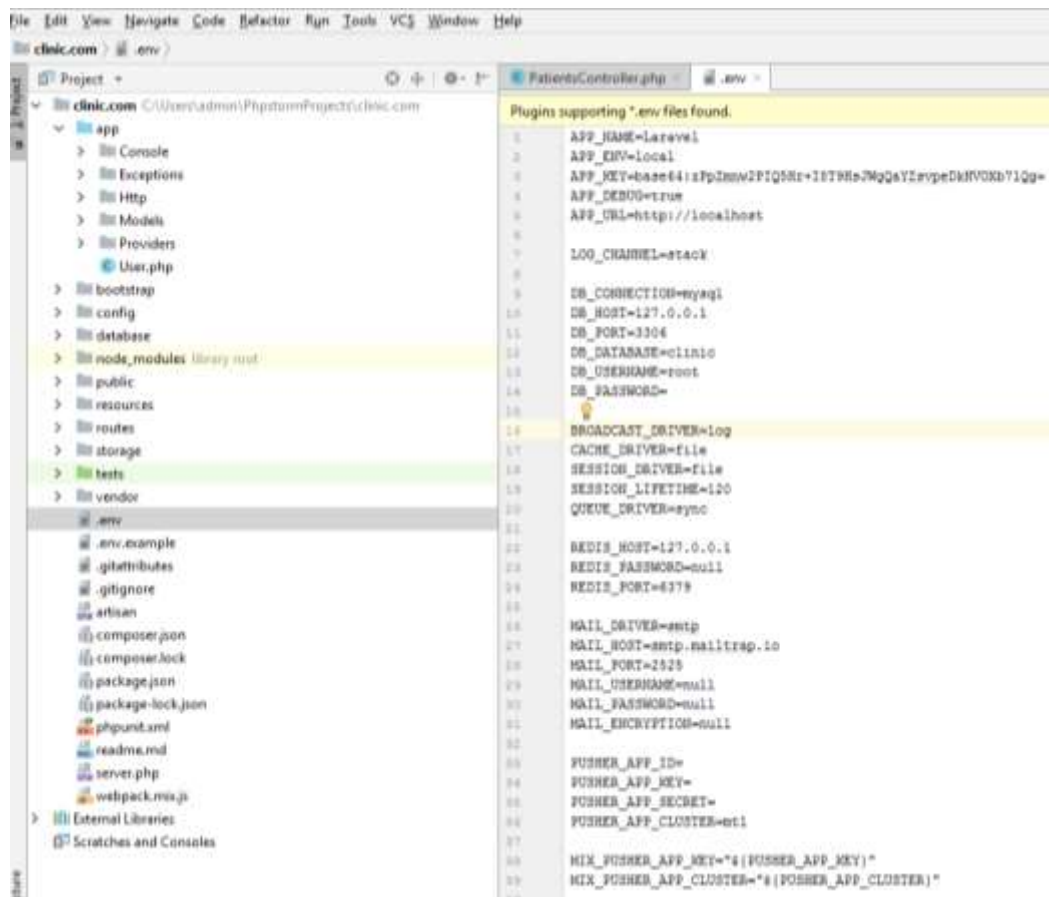


Рисунок 4.3 – З'єднання проекту та БД

Після цього необхідно в PHPStorm та виконати в терміналі команду – `php artisan migrate`. Дана команда виконає міграцію БД (рис. 4.4).

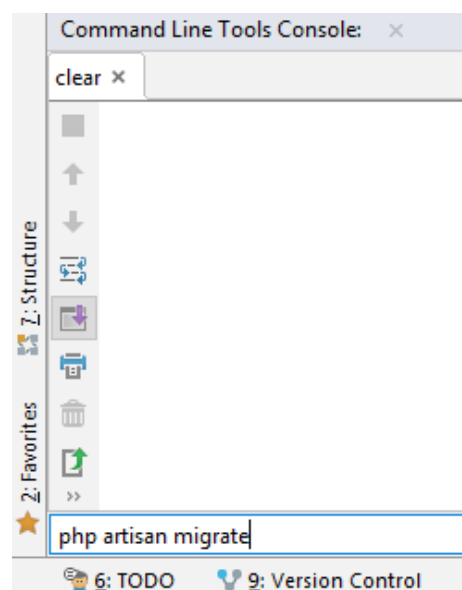


Рисунок 4.4 – Міграція БД

Далі необхідно запустити сервер. Для цього відкриваємо XAMPP та натискаємо кнопку «Start» навпроти Apache та MySQL (рис. 4.5).

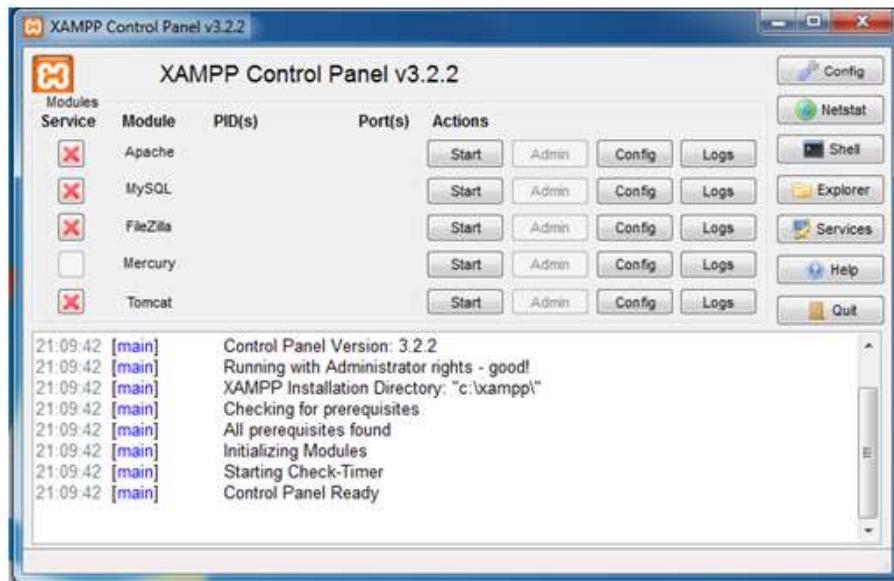


Рисунок 4.5 – Запуск серверу

Після цього можна ввести назву вебзастосунку в URL полі веб-браузера (рис. 4.6).

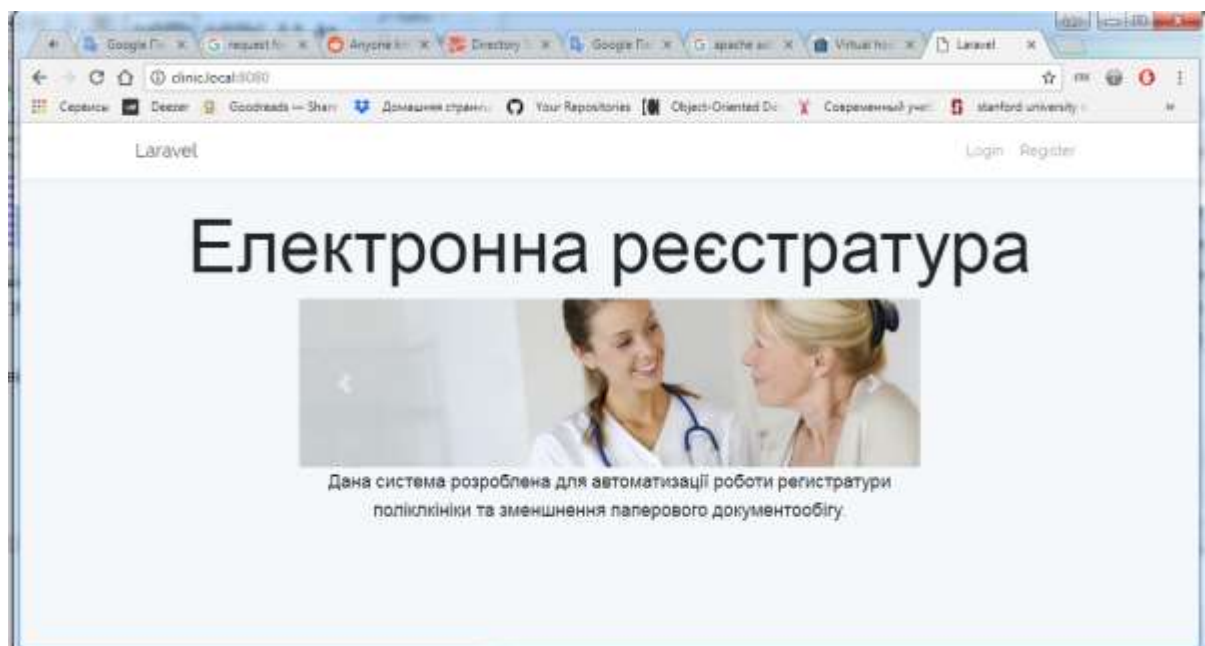


Рисунок 4.6 – Головна сторінка вебзастосунку

4.4 Повідомлення оператору

Під час роботи з програми користувач може отримати різні повідомлення. Основні з яких перелічені в таблиці 4.1.

Таблиця 4.1 – Основні повідомлення, які виникають при роботі із програмою

Повідомлення	Опис
«You cannot create this data»	Користувач не вповноважений відкрити форму створення запису
«You cannot store this data»	Користувач не вповноважений зберегти запис
«You cannot edit this data»	Користувач не вповноважений відкрити форму редагування запису
«You cannot update this data »	Користувач не вповноважений оновити запис
«You cannot destroy this data»	Користувач не вповноважений видалити запис
«Інформацію зареєстровано»	Виконано успішне зберігання
«Інформацію оновлено»	Виконано успішне оновлення
«Інформацію видалено»	Виконано успішне видалення
«Заповніть поле»	Поле форми має бути заповненим

Наведені повідомлення можуть виникати як сигнал успішного виконання дії, так і внаслідок некоректних дій користувача.

4.5 Тестування програмного продукту

Тестування розробленого програмного забезпечення буде відбуватися за наступним сценарієм:

- користувач запускає веб-браузер Google Chrome і вводить у рядок пошуку clinic.local:8080 та натискає клавішу enter;

- користувач натискає кнопку зареєструватися;
- користувач додає нового пацієнта;
- користувач змінює відомості про пацієнта;
- користувач видаляє інформацію про пацієнта;
- користувач виходить з системи;
- користувач заходить від імені пацієнта;
- користувач переглядає інформацію про себе;
- користувач виходить з системи.

Спочатку тестування, як і було зазначено в методиці проведення тестування, намагаємося зареєструватися з хибними даними (рис. 4.7).

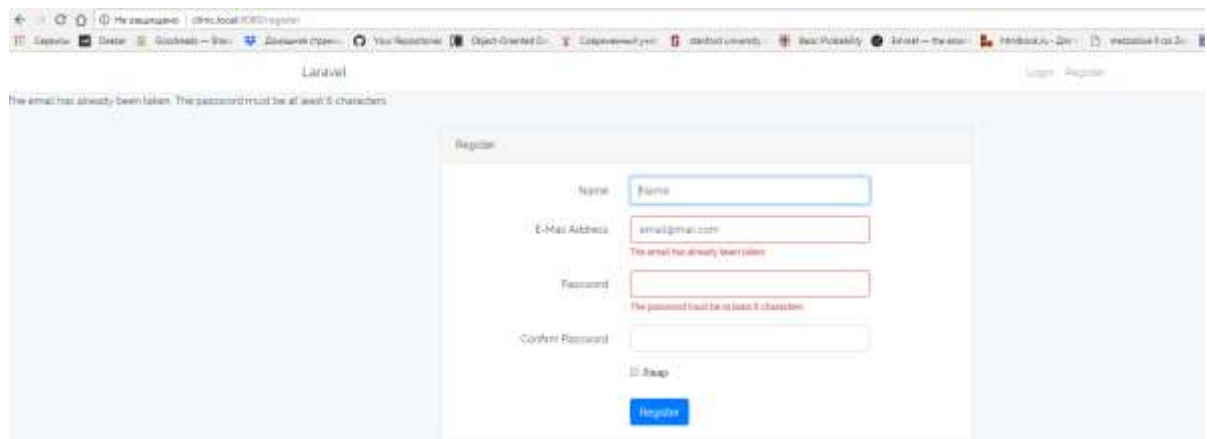


Рисунок 4.7 – Помилка при реєстрації

Тепер зареєструємося з вірними даними (рис. 4.8).

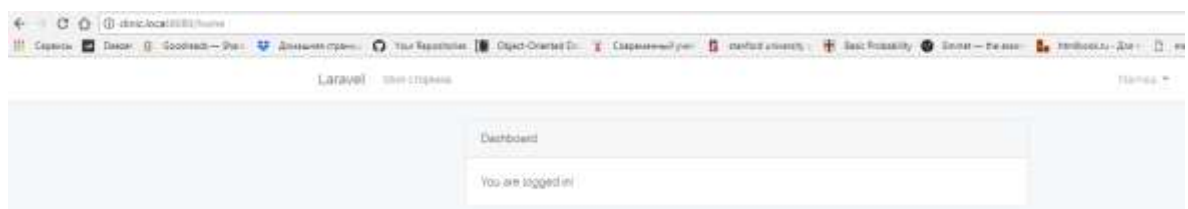


Рисунок 4.8 – Успішна реєстрація

Після реєстрації входимо до сторінки реєстрації пацієнта (рис. 4.9).

Рисунок 4.9 – Сторінка реєстрації пацієнта

Далі вводимо данні для реєстрації. Результат реєстрації зображений на рис. 4.10.

Код хворого	Ім'я	Провиде	По батькові	День народження	Диспансерна група
1	Name	Surname	Patronymic	2018-06-06	1
2	Name2	Surname2	Patronymic2	2018-06-06	0
3	Oleg	Iv	iv	2018-06-02	0
4	Петро	Петрович	Петрович	1990-06-06	0

Рисунок 4.10 – Результат реєстрації пацієнта

Після цього можна змінити відомості про пацієнта. Результат зміни відомостей зображено на рис. 4.11.

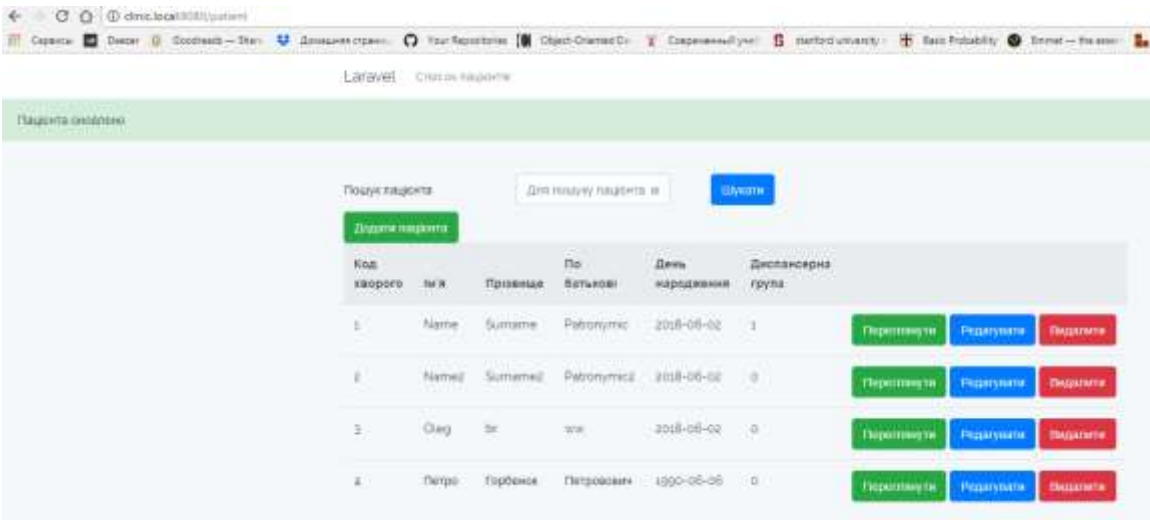


Рисунок 4.11 – Результат оновлення інформації про пацієнта

Далі заходимо в систему від імені пацієнта та переглядаємо інформацію (рис. 4.12).

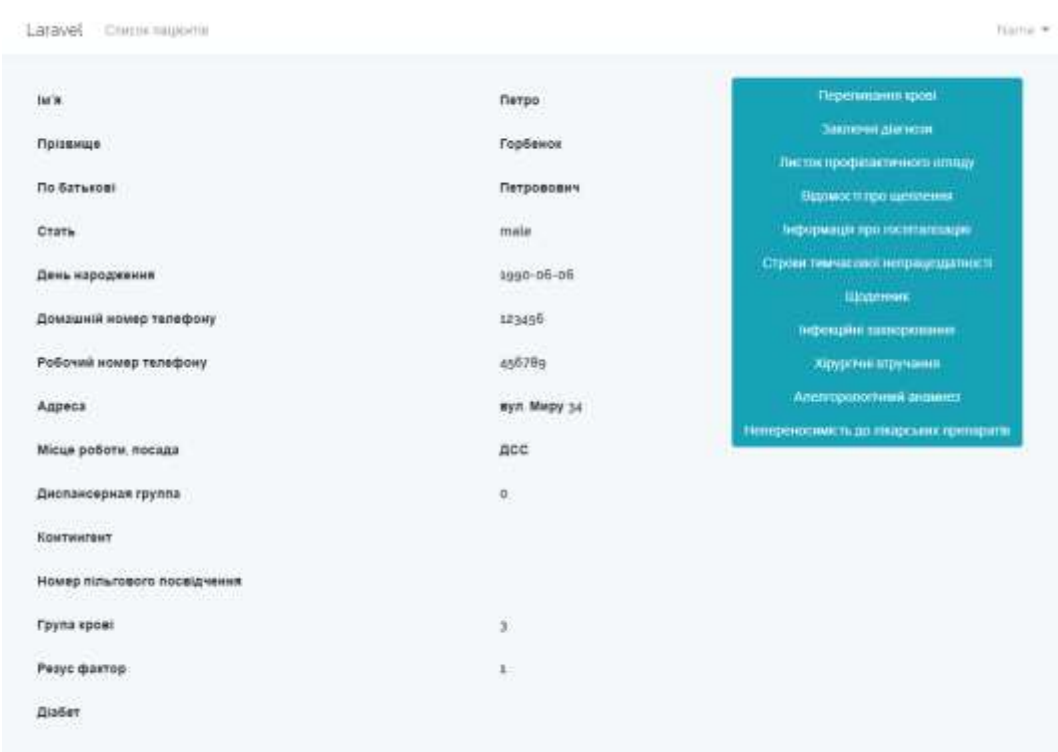


Рисунок 4.12 – Перегляд власної інформації пацієнтом

Останнім кроком буде перевірка механізму видалення інформації про пацієнта. Результат зображений на рис. 4.13.

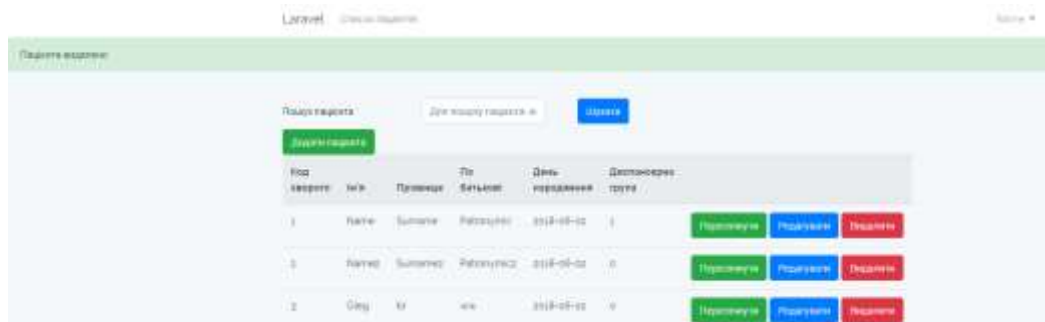


Рисунок 4.13 – Результат видалення пацієнта

Таким чином, результати тестування програми підтвердили її працездатність.

4.6 Висновки за розділом 4

Описано основні умови виконання програми. Виконано опис роботи користувача з програмним забезпеченням реєстратури центру надання медичної допомоги. Проведено тестування програмного забезпечення для реєстратури центру надання медичної допомоги. Результати тестування програми підтвердили, що розроблене програмне забезпечення реєстратури центру надання медичної допомоги може використовуватись за призначенням у медичних закладах.

5 КЕРІВНИЦТВО ПРОГРАМІСТА

5.1 Призначення й умови застосування програми

Програма призначена для ведення та зберігання ЕМК в медичних установах. Метою створення програми було зменшити паперовий документообіг в медичних установах та покращити якість діагностування пацієнтів, шляхом впровадження швидкого доступу до всієї необхідної інформації про хворого.

Програма виконує наступні функції:

- реєстрація користувачів;
- автентифікація користувачів;
- авторизація користувачів;
- CRUD операції з пацієнтами (права лікаря);
- перегляд персональної інформації (права пацієнта);
- пошук пацієнта за його прізвищем, ім'ям, по батькові.

Технічні засоби, за допомогою яких може експлуатуватися програма, повинні включати такі, що наведені нижче. У клієнта повинна бути наявність на пристрої доступу до Інтернет мережі та сучасного веб-браузера.

До складу технічних засобів повинен входити сервер, що задовольняє наступним вимогам:

- процесор з тактовою частотою не менше 2 GHz;
- оперативна пам'ять об'ємом 2 Gb, або більше;
- накопичувач об'ємом не менше 120 Gb;
- підтримка сервером MySQL версії не нижче 5.0;
- підтримка Apache HTTP Server, або nginx, або аналогічного веб-серверу;
- підтримка PHP версії не нижче 7.2.

5.2 Характеристики програми

Програма розроблена в середовищі PHPStorm за допомогою мови програмування PHP та фреймворку Laravel з використання MySQL у якості СКБД.

Час відновлення після відмови, викликаній збоєм електропостачання або зовнішніми факторами, не перевищує декількох хвилин, які необхідні для завантаження сервера та ресурсів, за умови дотримання умов експлуатації.

Відмови програми внаслідок некоректних дій користувача під час взаємодії користувача неприпустимі. У разі виникнення непередбачуваної події буде записаний та збережений лог. Якщо користувач матиме намір здійснити заборонені дії, він буде повернений на головну сторінку з повідомлення про неможливість виконання запиту.

5.3 Звернення до програми

Для запуску застосунку на локальному сервері з ОС Windows та встановленою збіркою веб-сервера XAMPP необхідно виконати наступні дії.

Створити віртуальний хост. Для цього відкривається конфігураційний файл Apache, який називається `httpd-vhosts.conf`, де визначається розташування проекту з вебзастосунком та назва за якою його можна знайти в веб-браузері (рис. 5.1).

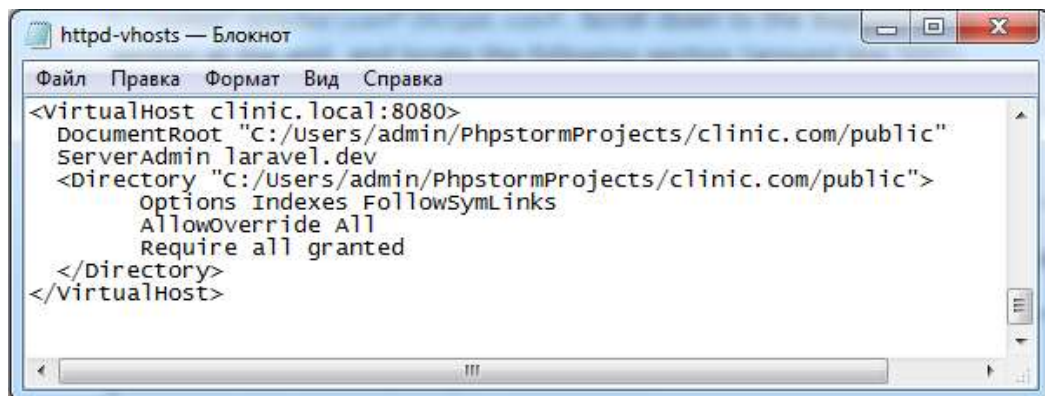


Рисунок 5.1 – Конфігураційний файл `httpd-vhosts.conf`

Після цього необхідно зарезервувати ім'я вебзастосунку на локальній машині. Для цього в файлі `C:\WINDOWS\system32\drivers\etc\hosts` вписується локальна адреса та резервоване ім'я (рис. 5.2).

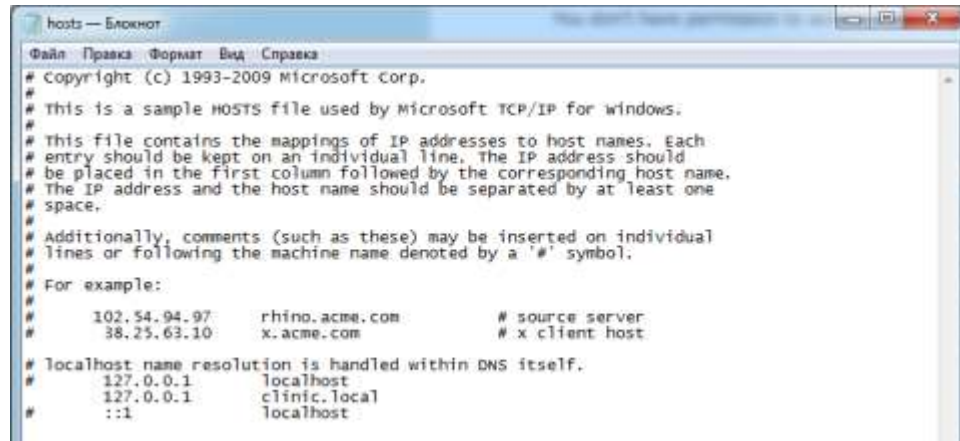


Рисунок 5.2 – Налаштування файлу hosts

Далі необхідно відкрити проект в PHPStorm та вказати з'єднання до БД (рис. 5.3).

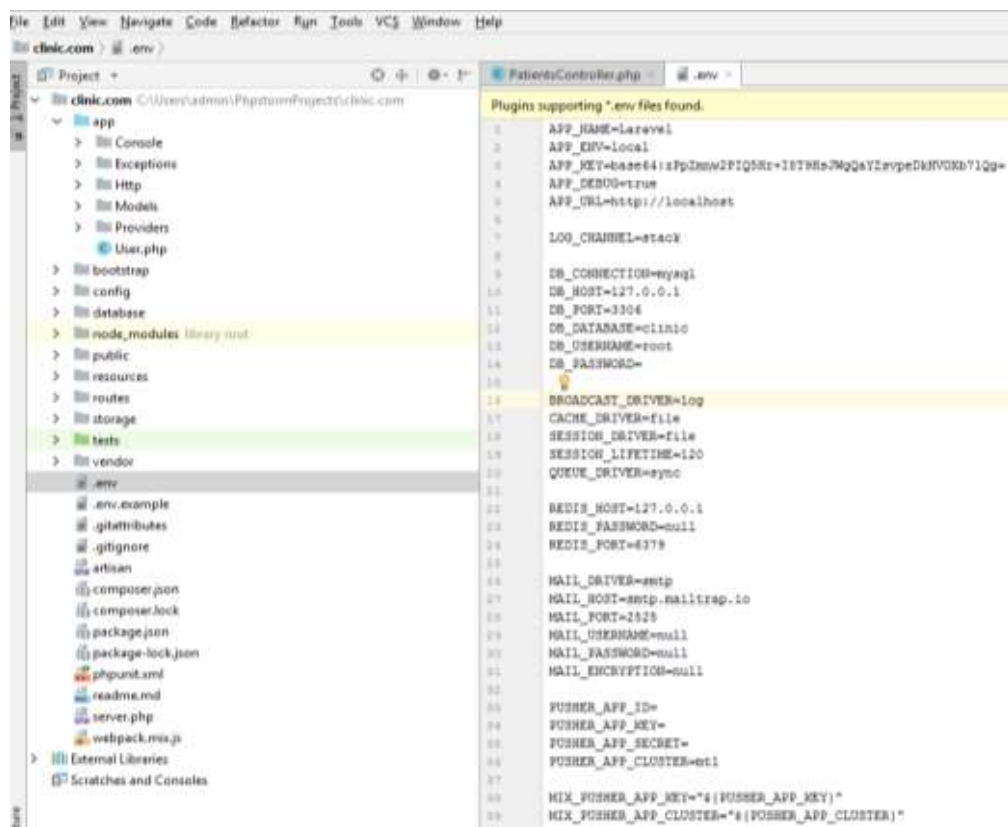


Рисунок 5.3 – З'єднання проекту та БД

Після цього необхідно RHPStorm та виконати в терміналі команду – `php artisan migrate`. Дана команда виконає міграцію БД (рис. 5.4).

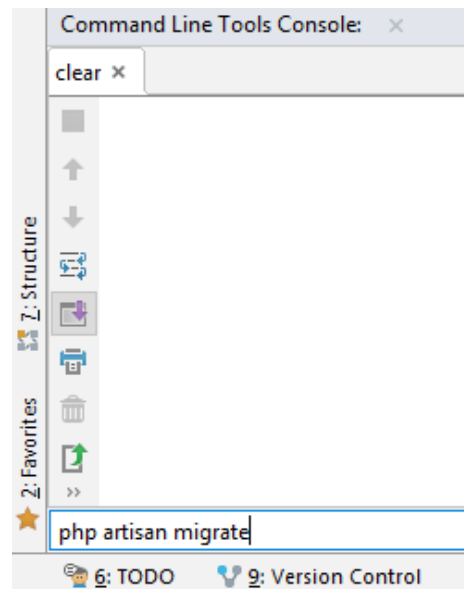


Рисунок 5.4 – Міграція БД

Далі необхідно запустити сервер. Для цього відкриваємо XAMPP та натискаємо кнопку «Start» навпроти Apache та MySQL (рис. 5.5).

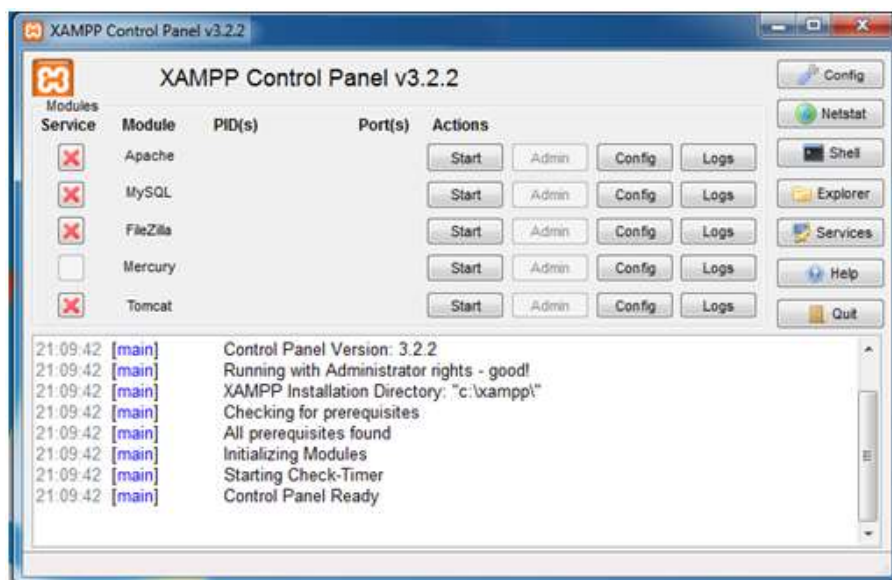


Рисунок 5.5 – Запуск серверу

Після цього можна ввести назву вебзастосунку в URL полі веб-браузера (рис. 5.6).

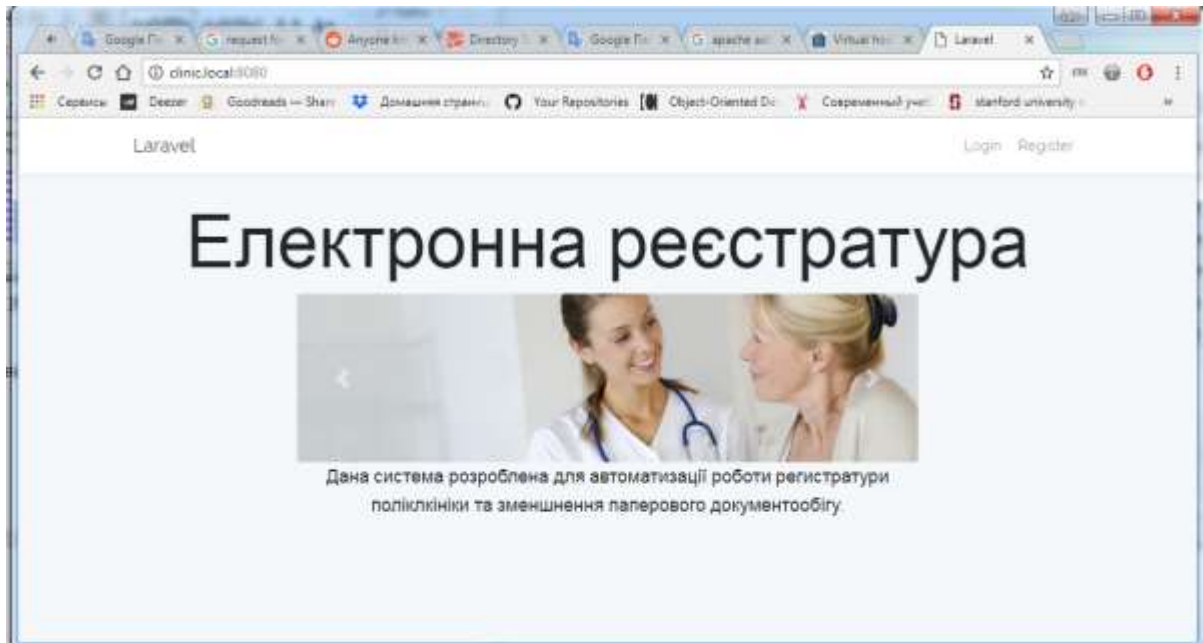


Рисунок 5.6 – Головна сторінка вебзастосунку

5.4 Вхідні дані

До вхідних даних програми відносяться запити користувача, форми та кнопки переходу між сторінками.

5.5 Вихідні дані

Вихідні дані програми організовані у вигляді інтерфейсу, що відображає форми для заповнення, таблиці та елементи керування.

ВИСНОВКИ

У дипломній кваліфікаційній роботі бакалавра виконано огляд проблеми автоматизації процесу документообігу в медичних установах. Проаналізовано програмне забезпечення для автоматизації документообігу. В результаті проведеного аналізу визначено, що доцільно розробити вебзастосунок, який буде орієнтований на зменшення паперового документообігу в реєстратурі медичного закладу, усунення недоліків існуючих програмних рішень та автоматизації роботи лікаря шляхом створення електронних документів.

Обґрунтовано вибір мови програмування PHP для розробки програмного забезпечення. Проаналізовано її переваги та недоліки, проведено порівняльний аналіз з мовами JavaScript та Java. Мову PHP обрано, враховуючи її переваги щодо зручності розробки, об'єктно-орієнтованих можливостей та часу створення вебзастосунків.

Для розробки програмного забезпечення обрано фреймворк Laravel, який є безкоштовним, з відкритим кодом PHP-фреймворком, призначеним для розробки вебзастосунків відповідно до шаблону MVC.

Проведено аналіз середовищ розробки Eclipse, PHPStorm, NetBeans. У результаті якого було визначено, що PHPStorm найкраще підходить для створення вебзастосунків за допомогою мови програмування PHP.

Визначено загальну структуру програми та побудовано архітектуру системи, яка відповідає меті роботи, та може реалізувати обрані функції за допомогою раніше визначених інструментів (програмних продуктів). Розроблено структурну схему програми у вигляді набору компонентів та зв'язків між ними. Проаналізовано основні вимоги до програмного забезпечення реєстратури центру надання медичної допомоги. Наведено функціональну схему програмного забезпечення реєстратури центру надання медичної допомоги. Описані основні компоненти побудованого програмного забезпечення за допомогою мови програмування PHP та фреймворку Laravel.

Розроблено програмне забезпечення реєстратури центру надання медичної допомоги. Описано основні умови виконання програми. Виконано опис роботи користувача з програмним забезпеченням реєстратури центру надання медичної допомоги. Проведено тестування програмного забезпечення для реєстратури центру надання медичної допомоги. Результати тестування програми підтвердили, що розроблене програмне забезпечення реєстратури центру надання медичної допомоги може використовуватись за призначенням у медичних закладах.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Електронний документообіг допомагає медзакладу заробляти [Електронний ресурс]. – Режим доступу : <https://www.vz.kiev.ua/elektronnyj-dokumentoobig-dopomagaye-medzakladu-zaroblyaty/>.
2. Електронний документообіг: передумови, складнощі та необхідні кадрові ресурси [Електронний ресурс]. – Режим доступу : <http://ukrpohliad.org/economics/elektronnyj-dokumentoobig-peredumovy-skladnoshhi-ta-neobhidni-kadrovi-resursy.html>.
3. 5 причин, чому паперовий документообіг у минулому [Електронний ресурс]. – Режим доступу : <https://medoc.ua/blog/5-prichin-chomu-paperovij-dokumentoobig-u-minulomu>.
4. ТОП 10 причин запровадити електронний документообіг [Електронний ресурс]. – Режим доступу : https://www.pdb.lviv.ua/news_edo1.
5. МОЗ скасувало морально застарілі форми первинної облікової документації [Електронний ресурс]. – Режим доступу : <https://moz.gov.ua/article/news/moz-skasuvalo-moralno-zastarili-formi-pervinnoi-oblikovoi-dokumentacii>.
6. Мій МедКабінет [Електронний ресурс]. – Режим доступу : <https://mymedcabinet.com.ua/>.
7. Медична система helsi.me [Електронний ресурс]. – Режим доступу : <https://helsi.me/>.
8. Департамент охорони здоров'я Запорізької міської ради [Електронний ресурс]. – Режим доступу : <http://health.zp.ua/>.
9. Deployment Patterns (Microsoft Enterprise Architecture, Patterns, and Practices) [Electronic resource]. – Access mode : [https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ff646997\(v%3dpandp.10\)](https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ff646997(v%3dpandp.10)).
10. PHP – довідник з PHP [Електронний ресурс]. – Режим доступу : <http://www.php.su/php/?history>.
11. Офіційна документація з PHP [Електронний ресурс]. – Режим

доступу : <http://php.net/manual/en/intro-what-is.php>.

12. Особливості PHP [Електронний ресурс]. – Режим доступу: <http://php.net/manual/en/features.php>.

13. Why Laravel is Taking the PHP Community by Storm – особливості Laravel [Electronic resource]. – Access mode : <https://code.tutsplus.com/tutorials/why-laravel-is-taking-the-php-community-by-storm--pre-52639>.

14. PhpStorm – IDE для PHP [Електронний ресурс]. – Режим доступу: <https://www.jetbrains.com/phpstorm/?fromMenu>.

ДОДАТОК А
Технічне завдання

Вступ

Для розв'язання проблеми забезпечення автоматизації реєстратури центру надання медичної допомоги запропоновано вебзастосунок управління електронною медичною карткою реєстратури центру надання медичної допомоги.

A.1 Підстави для розробки

Розробка виконується на основі завдання на дипломну кваліфікаційну роботу бакалавра, яке було визначене за темою “Програмне забезпечення реєстратури центру надання медичної допомоги” і затверджене наказом Національного університету «Запорізька політехніка».

A.2 Призначення розробки

Програмне забезпечення призначене для зменшення паперового документообігу в реєстратурах українських медичних закладах.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Функціональним призначенням продукту є надання представникам медичних закладів зручного інструменту для обліку пацієнтів. Функціональні характеристики, виконання яких має забезпечуватися програмою:

- коректна передача та прийом даних за допомогою веб-інтерфейсу;
- програма повинна мати «дружній» для користувача інтерфейс;
- розділення на дві ролі: лікар та пацієнт;

- лікарю доступні CRUD операції з будь-якою інформацією про пацієнта;
- пацієнту тільки перегляд персональної інформації;
- має бути можливість швидкого пошуку пацієнтів за ПІБ;
- розміри елементів управління повинні змінюватися відповідно до зміни розмірів робочого вікна;
- час обробки зображення не повинно перевищувати 10 секунд;
- час реакції програми на натискання будь-якої з клавіш і маніпуляції мишею не повинно перевищувати 1 секунди, в разі відповідності системних ресурсів вимогу до складу і параметрів технічних засобів.

A.3.2 Вимоги до надійності

Розроблювана система є web застосунком для медичних закладів, тому має підвищені вимоги щодо забезпечення безпеки даних користувачів та пацієнтів, для цього слід реалізувати захист від основних видів атак та вразливостей – CSRF (cross site request forgery), XSS (cross-site scripting), SQL (structured query language) injection та інших.

Відмови програми внаслідок некоректних дій користувача при роботі з інтерфейсом недопустимі. Слід забезпечити роботу користувачів з привілеями.

A.3.3 Умови експлуатації

Кліматичні умови експлуатації, при яких повинні забезпечуватися задані характеристики автоматизованої системи, повинні задовольняти вимогам, що пред'являються до технічних засобів у частині умов їх експлуатації.

Кінцевий користувач додатка повинен володіти навичками роботи з веб-інтерфейсом та вебзастосунками.

A.3.4 Вимоги до складу та параметрів технічних засобів

Для нормального функціонування системи на комп'ютері користувача, повинні виконуватися наступні умови:

- підтримка одного з сучасних веб-браузерів;
- доступ до Інтернет мережі.

Для нормального функціонування системи необхідний сервер, що задовольняє наступним вимогам:

- процесор з тактовою частотою не менше 2 GHz;
- оперативна пам'ять об'ємом 2 Gb, або більше;
- накопичувач об'ємом не менше 120 Gb;
- підтримка сервером MySQL версії не нижче 5.0;
- підтримка Apache HTTP Server, або nginx, або аналогічного веб-серверу;
- підтримка PHP версії не нижче 7.2.

A.3.5 Вимоги до інформаційної та програмної сумісності

Вихідні коди програми повинні бути Java#. В якості інтегрованого середовища розробки програми повинно бути використано середовище IntelliJ IDEA не нижче версії 2018.

Системні програмні засоби, що використовуються програмою повинні бути локалізовані на мові операційної системи.

A.3.6 Вимоги до програмної документації

Необхідно розробити програмні документи, що включають технічне завдання, текст програми, опис програми, що оформлюються додатками пояснювальної записки до дипломної кваліфікаційної роботи бакалавра.

А.3.7 Вимоги до маркування та упакування

Програмне забезпечення реєстратури центру надання медичної допомоги повинно мати маркування з позначенням найменування і дати виготовлення.

Диск з програмою разом з документацією укладається у відповідну тару згідно з вимог до пакування. Тара маркується згідно до вимог до маркування.

А.4 Порядок контролю та приймання

Дипломна кваліфікаційна робота бакалавра повинна виконуватися у повній відповідності з календарним планом, виконання етапів має контролюватися керівником роботи. Останім етапом є захист роботи перед екзаменаційною комісією.

ДОДАТОК Б
Текст програми

Текст файлу index.php

```
<?php
/**
 * Laravel - A PHP Framework For Web Artisans
 *
 * @package  Laravel
 * @author   Taylor Otwell <taylor@laravel.com>
 */

define('LARAVEL_START', microtime(true));

/*
|-----
|
| Register The Auto Loader
|-----
|
|
| Composer provides a convenient, automatically generated
class loader for
| our application. We just need to utilize it! We'll simply
require it
| into the script here so that we don't have to worry
about manual
| loading any of our classes later on. It feels great to
relax.
|
*/
require __DIR__.'/../vendor/autoload.php';
/*
|-----
|
| Turn On The Lights
|-----
|
|
| We need to illuminate PHP development, so let us turn on
the lights.
| This bootstraps the framework and gets it ready for use,
than it
```

```

| will load up this application so that we can run it and
sand
| the responses back to the browser and delight our users.
|
*/
$app = require_once __DIR__.'../bootstrap/app.php';
/*
|-----
-----
| Run The Application
|-----
-----
|
| Once we have the application, we can handle the incoming
request
| through the kernel, and send the associated response back to
| the client's browser allowing them to enjoy the creative
| and wonderful application we have prepared for them.
|
*/
$kernel = $app->make(Illuminate\Contracts\Http\Kernel::class);
$response = $kernel->handle(
    $request = Illuminate\Http\Request::capture()
);
$response->send();
$kernel->terminate($request, $response);

```

Текст файла AlergicHistoryController.php

```

<?php
namespace App\Http\Controllers;
use App\Models\AlergicHistory;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Gate;
class AlergicHistoryController extends Controller
{
    /**
     * Create a new Controller instance.
     *
     * @return void
     */
    public function __construct()

```

```

    {
        $this->middleware('auth');
    }
    /**
     * Display a listing of the resource.
     *
     * @param int $patientIDd
     * @return \Illuminate\Http\Response
     */
    public function index($patientIDd)
    {
        $alergiesList = AlergicHistory::where('patient_id',
$patientIDd)->get();
        return view('alergicHistories.history')-
>with(['alergiesList' => $alergiesList, 'patientIDd' =>
$patientIDd]);
    }
    /**
     * Show the form for creting a new resorce.
     *
     * @param int $patientIDd
     * @return \Illuminate\Http\Response
     */
    public function create($patientIDd)
    {
        if(Gate::allows('create-update-delete-actions')){
            return view('alergicHistories.registerAlergy')-
>with(['patientIDd' => $patientIDd]);
        }
        else
            return redirect('/')->with('error', 'You cann't
create alergic history');
    }
    /**
     * Store a newly created resorce in storage.
     *
     * @param \Illuminate\Http\Request $request
     * @param int $patientIDd
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request, $patientIDd)

```

```

{
    if (Gate::allows('create-update-delete-actions')) {
        $this->validate($request, [
            'alergyName' => 'required',
        ]);
        $newAlergy = new AlergicHistory();
        $newAlergy->patient_id = $patientIDd;
        $newAlergy->alergyName = $request-
>input('alergyName');
        $newAlergy->save();
        return redirect()-
>route('patient.alergicHistories.index', ['patient' =>
$patientIDd])->with('success', 'Хворобу зареєстровано');
    }
    else
        return redirect('/')->with('error', 'Ви не можете
зареєструвати');
}
/**
 *
 * @param int $id
 * @return \Illuminate\Http\Response
 */
public function show($id)
{
    //
}
/**
 * Show the form edditng the specifid resource.
 *
 * @param int $patientIDd
 * @param int $id
 * @return \Illuminate\Http\Response
 */
public function edit($patientIDd, $id)
{
    if (Gate::allows('create-update-delete-actions')) {
        $alergy = AlergicHistory::find($id);
        return view('alergicHistories.editAlergy')-
>with(['patientIDd' => $patientIDd, 'alergy' => $alergy]);
    } else {

```

```

        return redirect('/')->with('error', 'Ви не можете
редагувати');
    }
}
/**
 *
 * @param \Illuminate\Http\Request $request
 * @return \Illuminate\Http\Response
 */
public function update(Request $request, $patientIDd,
$id)
{
    if (Gate::allows('create-update-delete-actions')) {
        $this->validate($request, [
            'alergyName' => 'required',
        ]);
        $newAlergy = AlergicHistory::find($id);
        $newAlergy->alergyName = $request-
>input('alergyName');
        $newAlergy->save();
        return redirect()-
>route('patient.alergicHistories.index', ['patient' =>
$patientIDd])->with('success', 'Алергію оновлено');
    } else {
        return redirect('/')->with('error', 'Ви не можете
оновлювати дані');
    }
}
/**
 *
 * @param int $patientIDd
 * @param int $id
 * @return \Illuminate\Http\Response
 */
public function destroy($patientIDd, $id)
{
    if (Gate::allows('create-update-delete-actions')) {
        $alergy = AlergicHistory::find($id);
        $alergy->delete();
    }
}

```

```

                                return                                redirect() -
>route('patient.alergicHistories.index',          ['patient'          =>
$patientIDd])->with('success', 'Алергію видалено');
                                } else {
                                    return redirect('/')->with('error', 'Ви не можете
видаляти дані');
                                }
                            }
                    }
    }

```

Текст файлу PatientControler.php

```

<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
use App\Models\Patient;
use Illuminate\Support\Facades\Gate;
class PatientssControler extends Controller
{
    /**
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('auth');
    }
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        //TODO definitely use policies instead of gates
        if (Gate::allows('create-update-delete-actions')) {
            $Patientss = Patients::all();
            return view('Patientss.listOfPatients')-
>with('Patientss', $Patientss);
        } else {
            return redirect('/')->with('error', 'You can not
view all Patientss');

```

```

    }
}
/**
 * Show the form for creting a new resorce.
 *
 * @return \Illuminate\Http\Response
 */
public function create()
{
    if (Gate::allows('create-update-delete-actions')) {
        return view('Patientss.registerPatients');
    } else {
        return redirect('/')->with('error', 'You can not
create Patients');
    }
}
/**
 * Store a newly created resorce in storagge.
 *
 * @param \Illuminate\Http\Request $request
 * @return \Illuminate\Http\Response
 */
public function store(Request $request)
{
    if (Gate::allows('create-update-delete-actions')) {
        $this->validate($request, [//TODO improve
validation
        'nameFIO' => 'required',
        'surname_prizvyshche' => 'required',
        'patronimic' => 'required',
        'gender' => 'required',
        'birthdate' => 'required',
        'homePhnNmb' => 'required',
        'workPhnNmb' => 'required',
        'addr' => 'required',
        'whereHeWorks' => 'required',
        'groupOfdispensary' => 'required|boolean'
    ]);
    $newPatients = new Patients();
    $newPatients->nameFIO = $request->input('nameFIO');

```

```

        $newPatients->surname_prizvyshche = $request-
>input('surname_prizvyshche');
        $newPatients->patronimic = $request-
>input('patronimic');
        $newPatients->gender = $request->input('gender');
        $newPatients->birthdate = $request-
>input('birthdate');
        $newPatients->homePhnNmb = $request-
>input('homePhnNmb');
        $newPatients->workPhnNmb = $request-
>input('workPhnNmb');
        $newPatients->addr = $request->input('addr');
        $newPatients->whereHeWorks = $request-
>input('whereHeWorks');
        $newPatients->groupOfdispensary = ($request-
>input('groupOfdispensary') == 1 ? true : false);
        $newPatients->contingent = $request-
>input('contingent');
        $newPatients->PrivilegeCertificateID = $request-
>input('PrivilegeCertificateID');
        $newPatients->bloodType = $request-
>input('bloodType');
        $newPatients->rh = ($request->input('rh') == 1 ?
true : false);
        $newPatients->diabetes = $request-
>input('diabetes');
        $newPatients->save();
        return redirect()->route('Patients.index')-
>with('success', 'Додано нового пацієнта');
    } else {
        return redirect('/')->with('error', 'You can not
store Patients');
    }
}
/**
 *
 * @param int $id
 * @return \Illuminate\Http\Response
 */
public function show($id)
{

```

```

        if(auth()->user()->isDoctor){
            $Patients = Patients::find($id);
            return view('Patientss.Patients')-
>with('Patients', $Patients);
        }
        else{
            //TODO Patients can change PatientsIDd in url and
associate himself with other Patients
            $Patients = Patients::find($id);
            return view('Patientss.Patients')-
>with('Patients', $Patients);
        }
    }
    /**
     * Show the form for editing the specified resource.
     *
     * @param int $id
     * @return \Illuminate\Http\Response
     */
    public function edit($id)
    {
        if (Gate::allows('create-update-delete-actions')) {
            $Patients = Patients::find($id);
            return view('Patientss.editPatients')-
>with('Patients', $Patients);
        } else {
            return redirect('/')->with('error', 'You can not
edit Patients');
        }
    }
    /**
     *
     * @param \Illuminate\Http\Request $request
     * @param int $id
     * @return \Illuminate\Http\Response
     */
    public function update(Request $request, $id)
    {
        if (Gate::allows('create-update-delete-actions')) {
            $this->validate($request, [
                'nameFIO' => 'required',

```

```

        'surname_prizvyshche' => 'required',
        'patronimic' => 'required',
        'gender' => 'required',
        'birthdate' => 'required',
        'homePhnNmb' => 'required',
        'workPhnNmb' => 'required',
        'addr' => 'required',
        'whereHeWorks' => 'required',
        'groupOfdispensary' => 'required'
    ]);
    $newPatients = Patients::find($id);
    $newPatients->nameFIO = $request-
>input('nameFIO');
    $newPatients->surname_prizvyshche = $request-
>input('surname_prizvyshche');
    $newPatients->patronimic = $request-
>input('patronimic');
    $newPatients->gender = $request->input('gender');
    $newPatients->birthdate = $request-
>input('birthdate');
    $newPatients->homePhnNmb = $request-
>input('homePhnNmb');
    $newPatients->workPhnNmb = $request-
>input('workPhnNmb');
    $newPatients->addr = $request->input('addr');
    $newPatients->whereHeWorks = $request-
>input('whereHeWorks');
    $newPatients->groupOfdispensary = ($request-
>input('groupOfdispensary') == 1 ? true : false);
    $newPatients->contingent = $request-
>input('contingent');
    $newPatients->PrivilegeCertificateID = $request-
>input('PrivilegeCertificateID');
    $newPatients->bloodType = $request-
>input('bloodType');
    $newPatients->rh = ($request->input('rh') == 1 ?
true : false);
    $newPatients->diabetes = $request-
>input('diabetes');
    $newPatients->save();
    redirect('/');

```

```

        return redirect()->route('Patients.index')-
>with('success', 'Пацієнта оновлено');
    } else {
        return redirect('/')->with('error', 'You can not
update Patients');
    }
}
/**
 *
 * @param int $id
 * @return \Illuminate\Http\Response
 */
public function destroy($id)
{
    if (Gate::allows('create-update-delete-actions')) {
        $Patients = Patients::find($id);
        $Patients->delete();
        return redirect()->route('Patients.index')-
>with('success', 'Пацієнта видалено');
    } else {
        return redirect('/')->with('error', 'You can not
destroy Patients');
    }
}
/**
 * Set user_id to Patients.
 *
 * @param \Illuminate\Http\Request $request
 * @return \Illuminate\Http\Response
 */
public function setUserId(Request $request)
{
    $this->validate($request, [
        'PatientsIDd' => 'required|numeric',
    ]);
    $Patients = Patients::find($request-
>input('PatientsIDd'));
    if($Patients != null){
        $Patients->user_id = auth()->user()->id;
        $Patients->save();
    }
}

```

```

        return redirect()->route('Patients.show',
['Patients' => $Patients->id])->with('success', 'Ви вийшли до своєї
персональної сторінки');
    }
    else{
        return redirect('/')->with('error', 'Не існує
такого номера пацієнта');
    }
}
/**
 * Search Patients by his nameFIO, surname_prizvyshche
and patronimic.
 *
 * @param \Illuminate\Http\Request $request
 * @return \Illuminate\Http\Response
 */
public function searchPatients(Request $request){
    $this->validate($request, [
        'searchPatients' => 'required',
    ]);
    if (Gate::allows('create-update-delete-actions')) {
        $Patients = Patients::search($request-
>input('searchPatients'))->get();
        if($Patients != null)
            return view('Patientss.listOfPatients')-
>with('Patientss', $Patients);
        //echo $Patients;
    } else
        return redirect()->route('Patients.index')-
>with('error', 'Пацієнта не знайдено.');
```

```

    } else {
        return redirect('/')->with('error', 'You can not
search Patientss!');
```

```

    }
}
}

```

ДОДАТОК В
Слайди презентації

Національний університет «Запорізька політехніка»
Кафедра програмних засобів

Програмне забезпечення реєстратури центру надання медичної допомоги

Виконав

ст. гр. КНТ-118сп

Є. О. Агєєв

Керівник:

к.т.н., доцент

С. Ю. Скрупський

Рисунок В.1 – Слайд 1

Об'єкт, предмет та мета роботи

Об'єкт дослідження – процес реєстрації медичних даних.

Предмет дослідження – програмне забезпечення для керування електронними медичними картками.

Мета роботи – розроблення вебзастосунку для керування електронними картками в медичних установах.

Рисунок В.2 – Слайд 2

Завдання роботи

Для досягнення поставленої мети необхідно було розв'язати такі завдання:

- проаналізувати предметну область;
- обрати інструментарій розробки ПЗ;
- розробити структурну схему програми;
- розробити ПЗ;
- підготувати програмну документацію на розроблену програму;
- протестувати розроблену програму.

3

Рисунок В.3 – Слайд 3

Вибір мови програмування

Критерії порівняння	Мови програмування		
	PHP	JavaScript	Java
Можливість додати в HTML	+	+	+
Підтримка ООП	+	±	+
Підтримка прототипів	–	+	–
Підтримка процедурного програмування	+	–	–
Статична типізація	–	–	+
Динамічна типізація	+	+	–
Одночасне виконання паралельних задач	+	–	+

4

Рисунок В.4 – Слайд 4

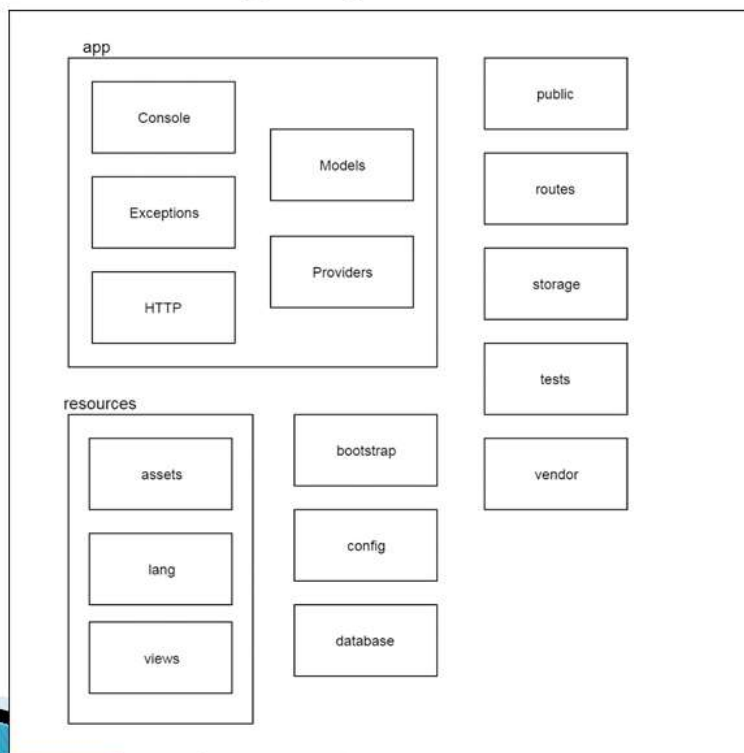
Обґрунтування вибору середовища розробки

Критерій порівняння	Eclipse	PHPStorm	NetBeans
Безкоштовність	+	±	+
Інтелектуальний рефакторинг	—	+	±
Розуміння контексту (autocomplete)	—	+	±
Пошук використання (Find Usage)	—	+	—
Кросплатформність	+	+	+
Вбудований дебагер	—	+	—

5

Рисунок В.5 – Слайд 5

Структурна схема компонентів програми



6

Рисунок В.6 – Слайд 6

Функціональна схема програми

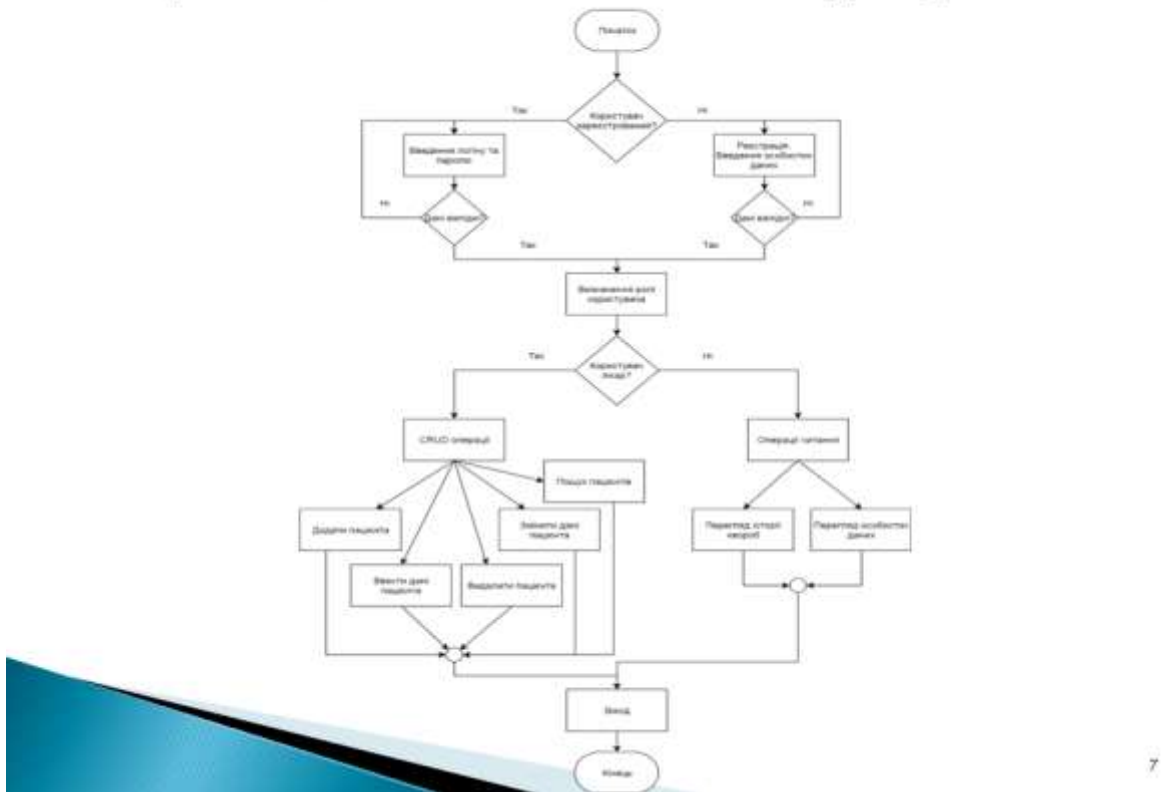


Рисунок В.7 – Слайд 7

Приклад роботи з програмою. Запуск на стороні сервера

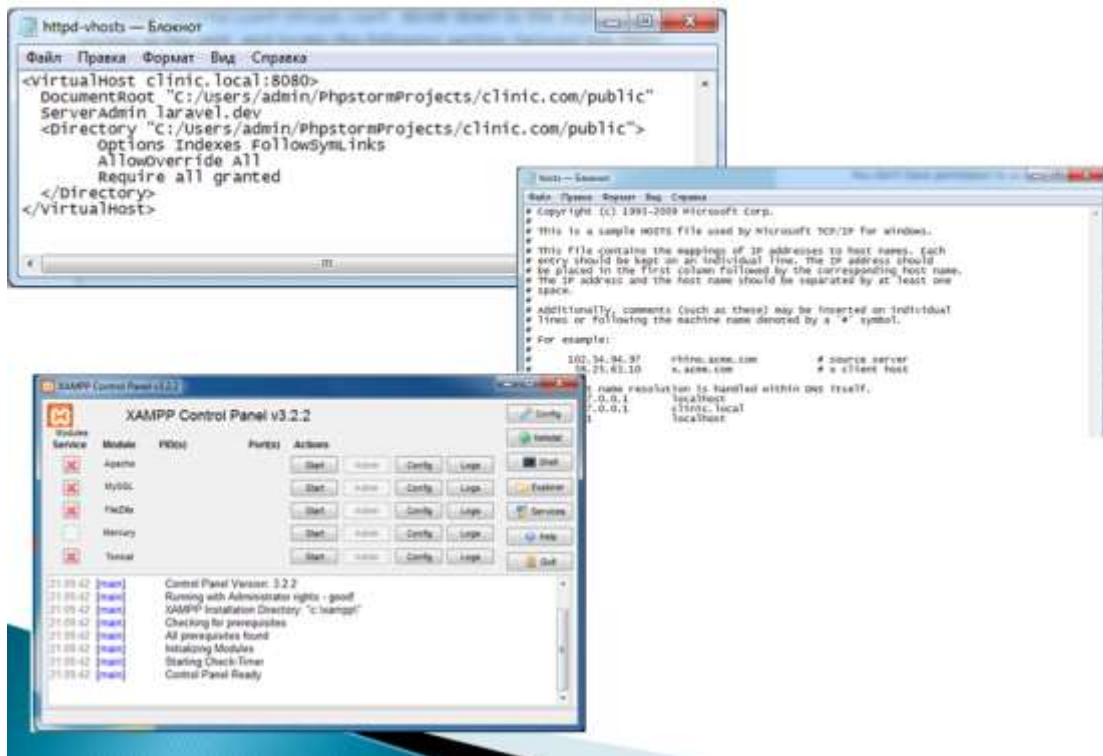


Рисунок В.8 – Слайд 8

Приклад роботи з програмою. Головна сторінка вебзастосунку

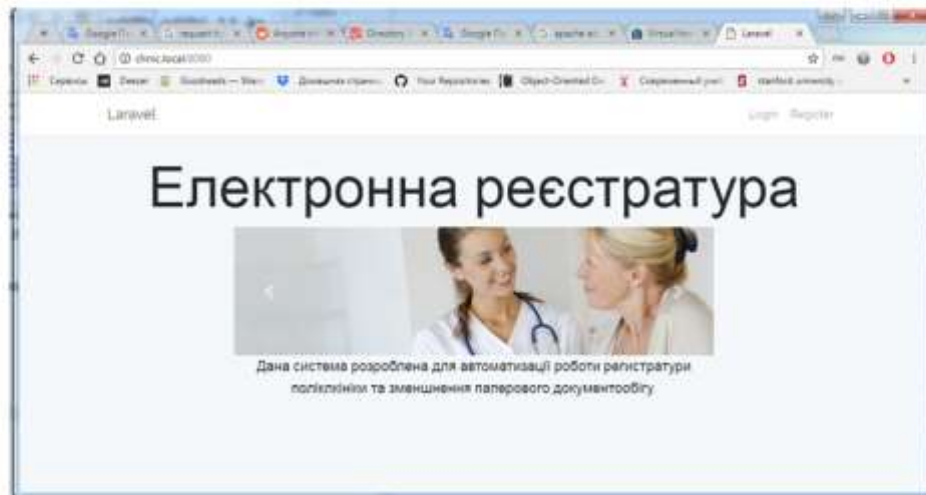


Рисунок В.9 – Слайд 9

Приклад роботи з програмою. Сторінка реєстрації

Рисунок В.10 – Слайд 10

ВИСНОВКИ

В процесі виконання роботи було створено програмне забезпечення реєстратури центру надання медичної допомоги.

Розв'язано такі завдання:

- аналіз предметної області;
- вибір інструментарію розробки ПЗ;
- розробка структурної схеми програми;
- розробка програмного забезпечення;
- підготовка програмної документації на розроблену програму;
- тестування розробленої програми.

13

Рисунок В.13 – Слайд 13

ВИСНОВКИ

Обґрунтовано вибір мови програмування PHP для розробки програмного забезпечення. Мову PHP обрано, враховуючи її переваги щодо зручності розробки, об'єктно-орієнтовних можливостей та часу створення вебзастосунків. Для розробки програмного забезпечення обрано фреймворк Laravel, який є безкоштовним, з відкритим кодом PHP-фреймворком, призначеним для розробки вебзастосунків відповідно до шаблону MVC.

Визначено загальну структуру програми та побудовано архітектуру системи. Розроблено структурну схему програми у вигляді набору компонентів та зв'язків між ними.

Розроблено програмне забезпечення реєстратури центру надання медичної допомоги. Виконано опис роботи користувача з програмним забезпеченням реєстратури центру надання медичної допомоги. Проведено тестування програмного забезпечення. Результати тестування програми підтвердили, що розроблене програмне забезпечення реєстратури центру надання медичної допомоги може використовуватись за призначенням у медичних закладах.

14

Рисунок В.14 – Слайд 14