Міністерство освіти і науки України
Сумський державний університет
Навчально-науковий інститут бізнес-технологій «УАБС»
Кафедра економічної кібернетики

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

на тему «Веб-орієнтована інформаційна система Електронна медична картка пацієнта»

Виконав студент 2 курсу, групи ЕК.м-61а
(номер курсу) (шифр групи)
Спеціальності 051«Економіка («Економічна кібернетика»)»
Беркут І.В.
(прізвище, ініціали студента)
Керівник к.т.н., доцент Гриценко К.Г.
(посада, науковий ступінь, прізвище, ініціали)

Суми – 2018 рік

ЗМІСТ

ВСТУП

Електронна медична картка – медична картка пацієнта медзакладу в електронній (безпаперовій) формі. Складається і зберігається в автоматизованій інформаційній базі даних медичної установи. Медичні картки мають поступово замінити традиційні картки медичного страхування.

Об’єктом дослідження є процес реєстрації пацієнтів в базі медичного закладу.

Предмет дослідження – автоматизація процесу реєстрації пацієнтів в базі медичного закладу.

Мета роботи полягає в розробці автоматизованої системи реєстрації пацієнтів в базі медичного закладу.

Для досягнення поставленої мети необхідно виконати наступні задачі:

–охарактеризувати основні принципи реєстрації пацієнтів;

–проаналізувати стан автоматизації в медичній сфері;

– сформулювати основні вимоги до автоматизованої системи реєстрації пацієнтів;

– обрати архітектуру та технологію для розробки автоматизованої системи;

–визначити склад функціональної частини автоматизованої системи;

– визначити склад підсистем функціональної частини автоматизованої системи;

–визначити програмне та апаратне забезпечення вирішення задачі;

– розробити структуру та визначити особливості реалізації інформаційного забезпечення системи;

– розробити структуру та визначити особливості реалізації алгоритмічного забезпечення;

–спроектувати інтерфейс користувача.

РОЗДІЛ 1 МЕДИЧНА КАРТКА ПАЦІЄНТА ЯК ОБ'ЄКТ АВТОМАТИЗАЦІЇ

1.1 Загальна характеристика електронної медичної картки пацієнта як об'єкта автоматизації

На даний час в світі набуває поширення автоматизація різних сфер промисловості. Не залишається без уваги і медицина. Протягом восьми останніх років в Україні спостерігається позитивна динаміка розвитку ринку медичних інформаційних систем. Великою проблемою на шляху до автоматизації системи охорони здоров'я є недостатнє фінансування, нерозуміння керівництвом можливостей, які дають сучасні технології для спрощення роботи медичних закладів.

Також існують і інші проблеми з якими стикається система охорони здоров'я, такі як: малий бюджет, підвищення вимог до системи охорони здоров'я, часта зміна місця проживання пацієнтів та ін.

Використання інформаційних систем, які дозволять автоматизувати роботи лікарень, значно підвищить не тільки рівень медичних послуг і якість лікування, а й ефективність використання медичних ресурсів

Незадовільною є ситуація з інформуванням міських управлінь охорони здоров'я, санепідемстанцій та інших установ про епідеміологічну ситуацію чи поточний стан захворюваності, та наявність вільних ліжок в лікарнях тощо.

Через відсутність сучасної техніки, програмного забезпечення та засобів зв'язку така інформація є неповною і запізнілою, що не дає можливості оперативно та адекватно попереджати загрози, а також реагувати на проблеми, які виникають у роботі медичних закладів.

Електронна медична карта пацієнта — це основний компонент інформаційної системи. У медичній карті зберігається вся інформація про пацієнта - реєстраційні дані, результати медичних оглядів, антропометричні

виміри, лабораторні обстеження і різноманітні графічні дані (УЗД, рентген і так далі).

Створення електронної медичної карти пацієнта, значно поліпшить та прискорить роботу медичних установ.

1.2 Аналіз стану впровадження електронних медичних карток пацієнтів

Більшість медичних інформаційних систем, які функціонують у лікувальних закладах в даний час, є морально і фізично застарілими. Переважно вони розроблені ще 10-15 років тому, їх ніхто вже давно не підтримує і не удосконалює. Ці системи дозволяють автоматизувати тільки підготовку звітних форм. На сьогодні на ринку медичних інформаційних системи присутні 10-15 розробників. За кількістю впроваджень слід відзначити: «Медсистеми», CIET, «Укрмедсоф», TherDep. До українського ринку проявляють інтерес також польські (ABG), російські(«Медиалог») та турецькі розробники медичних інформаційних систем. Проте вартість впровадження цих систем є значно вищою, ніж у аналогічних українських систем. Більшість систем побудовано на основі клієнт-серверної архітектури, яка забезпечує обмежену кількість функцій – переважно підготовку статистичних звітів та стандартних форм МОЗ. У цих системах ведеться електронна історія хвороби, внесення даних до яких здійснюється шляхом набору тексту або вибору фраз з довідників. Такий підхід не дає можливості в подальшому здійснювати поглиблений аналіз. Недоліком цих систем є необхідність звертатися до розробників для внесення змін у вхідні й вихідні форми. Приємно відзначити появу на ринку вітчизняних розробників систем, які підтримують 3-рівневу архітектуру. Це «Доктор Елекс» та «ЕмсіМед». Ці системи орієнтовані не тільки на державні, але й на приватні медичні заклади. Вони забезпечують інтеграцію електронної карти пацієнта з різноманітним діагностичним обладнанням, а також забезпечують отримання даних безпосередньо з

лабораторних аналізаторів. Внесення в електронну історію хвороби медичних даних здійснюється на основі розроблених лікарями-експертами протоколів. Це відкриває широкі можливості для подальшого всестороннього аналізу всіх даних. В цих системах є конструктор вхідних і вихідних звітних форм; вони забезпечують можливість обміну шаблонами документів. На особливу увагу заслуговує медична інформаційна система «Доктор Елекс». Вона розроблена з врахуванням сучасних стандартів та принципів взаємосумісності медичних інформаційних систем. В основі системи лежить ідея побудови лікарських оглядів на базі деревовидних шаблонів оглядів. Система забезпечує всі інформаційні потреби лікувально- реабілітаційного та діагностичного процесів, науково-дослідної та навчально-методичної роботи. Робота над створенням інформаційної системи в ТзОВ «Елекс» розпочалася ще в 1990 році. Першою розробкою компанії у медичній галузі була система «Авалон», впроваджена у ряді медичних закладів України. Подальший досвід компанія «Елекс» отримала при розробці онкологічної системи для університету міста Тампа, Каліфорнія, США та великої системи для збору статистики з використанням стандарту HL7 для американського ринку. Підсумком усіх інновацій стала система «Доктор Елекс», розроблена на найновіших технологіях із урахуванням досвіду і знань, отриманих фахівцями компанії під час роботи над попередніми системами. МІС дає можливість вводити в оптимальній формі, зберігати та аналізувати не тільки основні дані пацієнта, зазвичай використовувані у реєстратурі, а й усю медичну документацію, таку як скарги, анамнез життя і захворювання, дані об'єктивного обстеження, функціональної та лабораторної діагностики, антропометрії, а також дані про лікарські призначення та їх виконання впродовж перебування у лікувальній установі. Основним компонентом зберігання даних пацієнтів в інформаційній системі є електронна медична карта, в якій накопичується вся інформація: дані лікарських оглядів, антропометричні виміри, дані відео контролю, щоденники динамічного спостереження стану пацієнта, виписки та результати обстежень інших клінік, мультимедійні дані (рентгенограми, проби письма, фото) та інші важливі дані

про пацієнтів. Основна медична інформація, така як дані лікарського огляду та результати лікування, вводиться в електронну карту згідно спеціально розробленої уніфікованої медичної термінології, яка організована у деревовидні шаблони огляду — ієрархічні структури, що складаються із примітивів, які формують логіку лікарського обстеження. Система пройшла незалежне тестування і рекомендується МОЗ до впровадження в медичних закладах. Впровадження інформаційних технологій в медицині заслуговує на безпосередню увагу керівників галузі і зацікавлених відомств. Одним з пріоритетних напрямів розвитку системи охорони здоров'я є створення єдиного медичного інформаційного простору, який забезпечить прийняття ефективних управлінських рішень на всіх рівнях. Це дасть можливість налагодити ефективний облік діяльності медичним закладам організації здійснювати на сучасному рівні менеджмент, своєчасно отримувати інформацію про передові досягнення в галузі медичної науки, використовувати всю медичну інформацію про пацієнта (за весь період його життя), накопичену зі всіх рівнів надання медичної допомоги для досягнення кращого лікувального ефекту.

1.3 Формування вимог до веб-орієнтованої інформаційної системи «Електронна медична картка пацієнта»

Незважаючи на велике розмаїття програмних систем, що застосовуються для керування контентом, актуальним завданням залишається дослідження, розробка та вдосконалення засобів універсального керування інформаційними об’єктами Web-ресурсів, що дозволить спростити створення нових ресурсів різного призначення, а також забезпечить ефективні механізми їх супроводження та налаштування. У навчальному виданні розглянуто ключові засади створення сучасних Web-орієнтованих інформаційних систем.

Розглянемо дану задачу для вирішення автоматизованої системи, яка реалізує інформаційну технологію виконання встановлених функцій за

допомогою персоналу і комплексу засобів а також, дасть людям змогу з реєстрації електронних медичних карток.

Найбільшою проблемою, яку треба подолати у цій автоматизованій системі, є проблема реєстрації пацієнтів в медичному закладі.

В цьому разі, програмний веб-додаток має бути зробленим у простій і водночас зрозумілій формі яку будуть використовувати для користування, що дозволить швидко орієнтуватись та працювати з веб-додатком.

Також, дуже важливою особливістю є крос-платформне програмування яке дасть змогу користуватися системою на різних браузерах. Крос-платформний вигляд веб додатку можна досягли лише за допомогою кросбраузерності css. Тим, хто знайомий з HTML, не потрібно розповідати, що це мова розмітки сторінки. Але його творці вирішили додати теги, що відповідають за зовнішній вигляд і дизайн. Ось тільки при такому підході код сторінки ставав занадто об'ємним і практично нечитабельним. Природно, цей шлях вів у нікуди, тому було прийнято комплексне рішення: HTML відповідає за розмітку сторінок, CSS – за візуальне оформлення. Блокова розмітка сторінки засобами CSS має кілька незаперечних переваг. По-перше, стиль об'єктів знаходиться окремо від HTML документа, що значно підвищує читабельність коду і дозволяє швидко проводити візуальні зміни. По-друге, є можливість накласти один шар на інший, а це в кілька разів полегшує процес позиціонування. Природно, такі сайти швидше завантажуються і індексуються пошуковими системами. Розмітка сторінки в CSS дозволяє легко задавати сучасні візуальні ефекти. Єдиний недолік такого підходу – різне «розуміння» браузерами. Деякі відображають ресурс в такому вигляді, в якому бачить його веб-майстер. Але є браузери, що викривляють зображення, тому при великій кількості класів та селекторів необхідно використовувати методи, що зроблять код кроссбраузерним.

Для збереження даних які користувач буде заповняти ми будемо використовувати MySQL.

Крім цього, завдяки використанню веб-додатку, лікар матиме змогу здійснювати реєстрацію нових пацієнтів зі свого смартфона, планшета, персонального комп’ютера, або ноутбука швидко та зручно, оскільки дані пристрої завжди біля користувача.

Отже, вимоги – це можливості або умови, яким повинна відповідати система чи проект. Основне завдання етапу визначення вимог – знайти, обсудити і зафіксувати, що дійсно вимагається від системи у формі, яка буде зрозуміла і клієнтам і розробникам.

Тобто, необхідно розробити перший пакет вимог до системи. Далі, у розробці та тестуванні веб-додатку потреби будуть уточнюватись і доповнюватись. Даний підхід допоможе максимально точно забезпечити виконання вимог користувача, і є також є із основних засад уніфікованого підходу. У проекті ми повинні стежити за стрункістю і мінімалізмом архітектури використовувавши DRY, KISS. Повинні писати як на сучасній мові програмування, поділяючи на публічні ресурси і файли програми. Завжди валідувати будь які користувальницькі введення, екрануючи висновок. Потрібно стежити за відсутністю сміття в коді і називати методи і змінні так, щоб по одній назві було зрозуміло що до чого.

Більш детальну інформацію про дані підходи та розробку можна знайти у наступних розділах даної роботи.

Отже, виходячи із бачення продукту, можна виділити ряд правил високого рівня. Ці правила забезпечують про якість програмного засобу. До них відносяться:

- програмний веб-додаток має виконувати свою головну функцію, а саме реєстрацію пацієнта медичної установи;
- дана система має бути зручна для користувача і задовольняти його потреби, пов'язані зі сферою роботи веб-додатку;
- веб-сайт має допомагати людям заощаджувати свій час, а також економічний стан.

Всі інші вимоги, які деталізують сам програмний продукт, можуть змінюватись у рамках уніфікованого підходу та ітеративної розробки. Отже, доцільним є розгляд вимог за певною системою. Однією із таких систем, яка гарно себе зарекомендувала у світовій практиці розробки програмних продуктів, є система FURPS+.

Суть її полягає в тому, що програмний додаток варто робити на функціональні вимоги, які пристосовуються до можливостей системи а також властивостей які у свою чергу поділяються на вимоги зручності, надійності, продуктивності, та можливості підтримки.

Отже, за даною моделлю було складено перший пакет вимог до програмного продукту:

- Вимоги що, до функціонування:
    - Ведення користувачем. Весь процес реєстрації має бути зручним для користувальницького введенням, щоб повною мірою контролювати користувача.
    - Персоналізація. Для кращої диференціації інформації потрібно створити систему персоналізації користувача.
    - Доступність і надійність особливо важливі для веб-додатків з великою кількістю користувачів. Додатки повинні бути здатні витримувати великі навантаження і зобов'язані працювати навіть в нестандартних ситуаціях.
    - Аналіз сайту. Адміністратор повинен мати можливість виконувати та переглядати проаналізовану інформацію у різних форматах.
    - Швидкість веб-додатка. Програма має швидко реагувати на різні запити.
    - Гнучкість системної архітектури. Вона має підтримувати можливість для змін у контенту структуру додатку без негативного впливу на другі частини даної структури.
- Зручність веб додатку:

- Юзабіліті означає, що користувач може працювати ефективно, зручно і при цьому ще отримувати задоволення від роботи програми. Доступність є частиною юзабіліті.
- Люди повинні легко розуміти, що їй потрібно зробити, щоб отримати результат без будь яких проблем.
- Повнота та зрозумілість.

- Вимоги надійності:
  - Частота збоїв. Частота збоїв має бути зведена до мінімуму, у разі виникнення помилки, користувачу має бути пред'явлена зрозуміла коротка інформація про проблему та інструкції щодо її подолання.
  - Стійкість бази даних. Процес взаємодії із базою даних має бути спроектований таким чином, щоб при жодних обставинах не відбувалася втрата інформації або блокування її.
- Вимоги продуктивності:
  - Час відгуку. Час відгуку має зведений до мінімуму, або не більше 100 мс. Для деяких операції припускається трохи збільшений час відгуку.
  - Точність. Програмна система має безпомилково записувати всі дані, які вказав користувач, та здійснювати їх аналіз.
  - Доступність. Робота з веб-додатком має бути доступна для користувача у будь-який час.
  - Використання ресурсів. веб-додаток повинен добре індексуватися гугл аналітикою.
- Вимоги можливості підтримки:
  - Конфігурування. Програмна система має підтримувати систему конфігурації, для досягнення кращої відповідності потребам користувача.
  - Он-лайн підтримка веб додатку. Для відповідей на питання, у програмі має бути зазначена електронна адреса розробників.

Основна мета забезпечення якості, а точніше, управління якістю програмного забезпечення, що - зробити витрати і вигоди прозорими для всіх сторін, що беруть участь у створенні програмного забезпечення. Низька якість програми в довгостроковій перспективі викликає додаткові витрати. Якщо ці витрати можуть бути визначені кількісно, то потрібно зробити висновок, що досягнення кращої якості призведе до зниження витрат в майбутньому. Це, мабуть, єдиний спосіб змусити керівництво або клієнта розглянути питання про виділення бюджету для рефакторинга коду.

Web-технологія повністю перевернула уявлення про роботу з інформацією, та й з комп'ютером взагалі. Виявилося, що традиційні параметри розвитку обчислювальної техніки - продуктивність, пропускна здатність, ємність запам'ятовуючих пристроїв - не враховували головного "вузького місця" системи - інтерфейсу з людиною. Застарілий механізм взаємодії людини з інформаційною системою стримував впровадження нових технологій і зменшував вигоду від їх застосування. І тільки коли інтерфейс між людиною і комп'ютером був спрощений до природності сприйняття звичайною людиною, по слідував безпрецедентний вибух інтересу до можливостей обчислювальної техніки.

Інформація, доступна користувачам інтернет, розташовується на Web-серверах. Значна частина цієї інформації організована у вигляді веб-сайтів. Кожен з них має свою адресу. Веб-сайт - це інформація, представлена в певному виді. Для перегляду веб-сайтів на комп'ютері користувача використовуються спеціальні програми, які називаються браузерами. Найбільш поширеними браузерами в даний час є Google Chrome, Opera Mozilla Firefox. В залежності від того, яку адресу ми задамо в рядку браузер буде завантажувати в своє вікно відповідну інформацію.

РОЗДІЛ 2 РОЗРОБКА ВЕБ-ОРІЄНТОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ «ЕЛЕКТРОННА МЕДИЧНА КАРТКА ПАЦІЄНТА»

2.1 Архітектура автоматизованої системи

Дивлячись на основні правила розробки веб додатку сформуєму наступну архітектуру автоматизованої системи.

Система буде розрозроблена за допомогою використання програмного коду PHP. PHP (Hypertext Preprocessor) — це мова програмування яка має велику популярність, має загальне призначення з відкритим кодом вихода. PHP орієнтується для розробки веб додатків. PHP додатки обробляються на сервері та генерують HTML, за допомогою настройки сервера файли HTML обробляються процесами PHP. Дана мова програмування проста для вивчення, але може задовольнити велику кылькысть проблем програмістів різних рівнів.

Для того, щоб розробка прототипу була простіша та швидша, використовуватимимо фреймворк Laravel – безкоштовний, з відкритим кодом PHP-фреймворк, створений Taylor Otwell і призначений для розробки веб-додатків відповідно до шаблону «Модель-вид-контролер» (MVC) (рисунок 2.1). 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, а також його орієнтація на синтаксичний цукор.

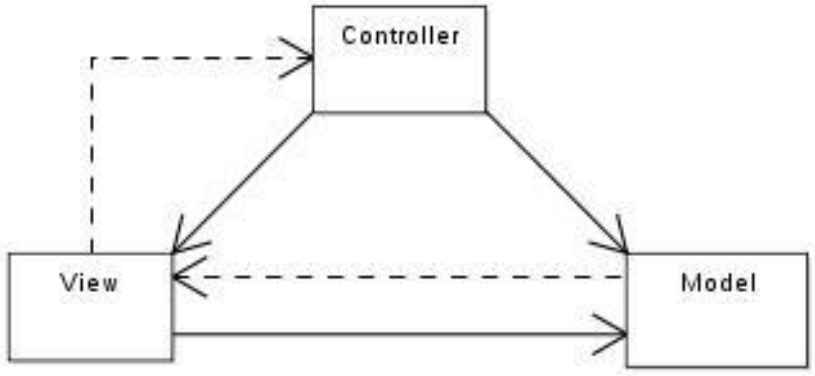


Рисунок 2.1 - патерн «Модель-вид-контролер»

Модель – це центральний компонент шаблону MVC який відображає поведінку застосунку, незалежну від інтерфейсу користувача. Модель стосується прямого керування даними, логікою та правилами застосунку.

Вид 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, наприклад гістограма для керівництва компанії й таблиці для бухгалтерії.

Контроллер – одержує вхідні дані й перетворює їх на команди для моделі чи вигляду.

Наприклад користувач потрапляє на головну сторінку сайту, реєструється. Допустимо, що користувач хоче подивитися дані, які вказав при реєстрації. Для перевірки інформації він обере відповідний розділ сайту. Контроллер отримає запит користувача, перевірить його а потім визве модель, запросивши в неї дані які влаштовують запит. Використавши ці дані він поверне інформацію до того шаблону який відповідає даній категорії.

Для розробки веб-додатку будемо використовувати операційну систему Linux Ubuntu. Це операційна система заснована Debian GNU/Linux. Ubuntu надає користувачу мінімальний набір програм загального призначення: багатовіконне стільничне середовище, засоби для перегляду Інтернету, організації електронної пошти, офісні програми з можливістю читати і записувати файли у форматах, що використовуються в пакеті програм Microsoft Office, редактор зображень, програвач компакт-дисків тощо. Спеціалізоване

програмне забезпечення, потрібне досвідченішим користувачам, можна отримати з відповідних репозиторіїв. Серверний варіант системи включає також засоби, потрібні для організації сервера баз даних, веб-сервера, сервера електронної пошти тощо.

Для розробки прототипу веб-орієнтованої інформаційної системи «електронна медична картка пацієнта» знадобиться наступне програмне забезпечення: Apache HHTP Server, MySql, PHP, PHPMyAdmin, PhpStorm, Composer.

В якості бази даних ми будемо використовувати MySql. MySql вільна та швидка реляційна база даних з відкритим кодом була створена як альтернатива комерційним системам. Дана база даних найкраще підходить до веб-додатків. Адже швидківсть обробки даних в о обсязі до 500000 записів є найкращою. В неї відкрита, вільна та безкоштовна ліцензія, а також найбільше підходить до більшості хостингових компаній в Україні. ЇЇ можна використовувати на різних платформах таких як Unix, Windows, інші. Кілька процесів можуть одночасно без жодних проблем читати дані з однієї бази. MySQL має подвійне ліцензування. MySQL може розповсюджуватися відповідно до умов ліцензії GPL. Однак за умовами GPL, якщо якась програма включає вихідні коди MySQL, то вона теж повинна розповсюджуватися за ліцензією GPL. Це може розходитися з планами розробників, не бажаючих відкривати вихідні тексти своїх програм. Для таких випадків передбачена комерційна ліцензія, яка також забезпечує якісну сервісну підтримку.

Apache HHTP Server - проект який розвивається The Apache Software Foundation, в результаті якого розробляється кроссплатформенний HTTP сервер з відкритим вихідним кодом. Будучи лідером найбільших вагомих на ринку розробнків Open Source, наштовхнула до прихильності компанії Apache Software Foundation, для того щоб створити постійне високоякісне програмне забезпечення, яке поліпшує майбутнє відкритої розробки. Apache підтримує половину інтернету, управляючи великою кількістю даних, виконує до 1000 мільярдів операцій в секунду, а також зберігає обєкти в кожній галузі.

Програмні проекти Apache є невід'ємною частиною майже будь-якого обчислювального пристрою кінцевого користувача, від ноутбуків до планшетів та телефонів. Код Apache для проектів складається більше ніж 6000 добровольців та співробітників корпорацій на шести континентах. Входить також в комплекс серверного програмного забезпечення як LAMP, а також популярного рішення для тих хто, потребує у веб сервері для отладки сценаріїв PHP, Perl і є найбільш зручний XAMPP. Apache можно реалізувати чотирма різними варіантами:

стандартні налаштування за замовчуванням в папці /var/www/html.

Доступ буде здійснений по адресу http://localhost/;

Налаштування основного хостингу, прикладом буде

http://localhost/phpmyadmin;

за допомогою віртуальних хостів в папці, приклад http://mysite/

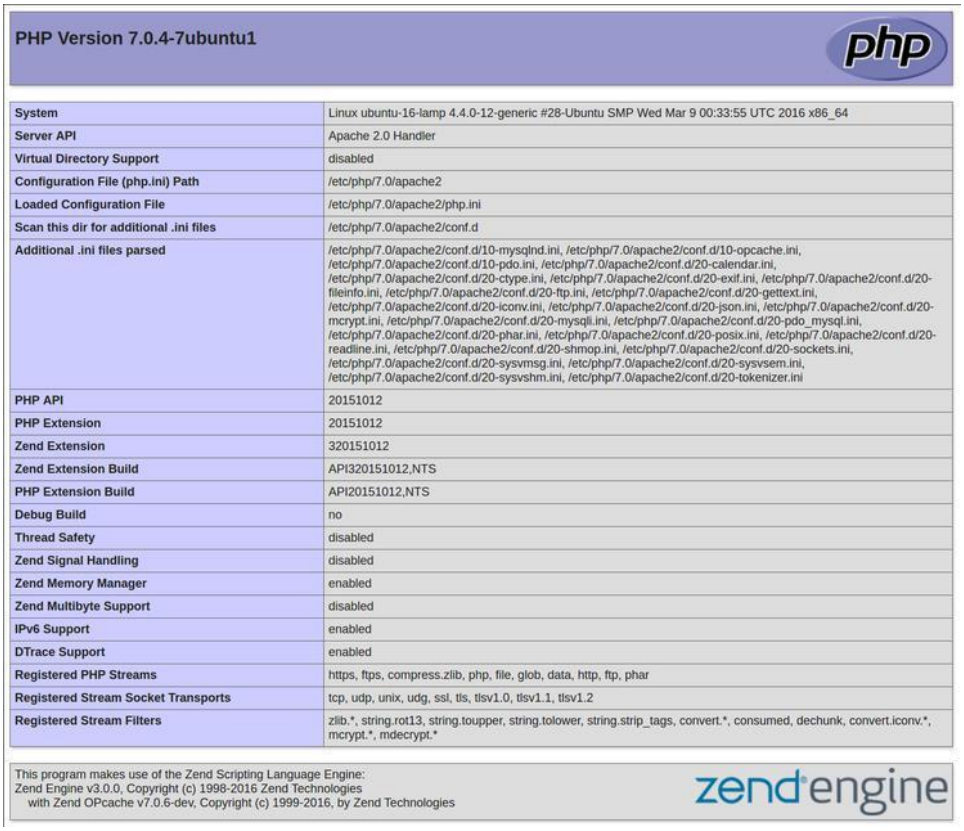
модуль userdir в папці користувача public_html, приклад

http://localhost/~username;

PHP це компонент який обробляє код, для виводу коненту. Він в загалом, може використовувати скріпти, підключатися до нашої бази даних MySQL, для того щоб ми змогли відобразити і передати контент в наш веб-сервіс. Потрібно встановити допоміжні менеджери пекетів, для того щоб PHP міг працювати з нашим сервером Apache і для того щоб MySQL без пороблем спілкувалося з PHP: php, libapache2-mod-php, php-mcrypt, php-mysql. Дана команда повинна встановити PHP без проблем.

Щоб Apache краще шукав файли PHP при заросі директорії, а не фали HTML (на даний час в першу чергу він буде шукати файл з назвою index.html), потрібно зробити наступне, в папці /etc/apache2/mods-enabled/ відкрити файл dir.conf і замінити місцями index.html з index.php. Після того як ми це зробимо, потрібно зберегти файл, та перезапустити сервер Apache. Щоб перевірити, що наша система PHP працює та сконфігурувалась, зробимо простий скріпт. Назвемо його info.php, та збережемо його в папці /var/www/html. Для перевірки

потрібно відкрити лише дану сторінку з веб-браузера. Сторінку яку ми побачимо приведена на рис 2.2 . Ця сторінка несе в собі інформацію про наш сервер з точки PHP. Вона є важливою для отладки та коректності прийнятих налаштувань.



PHP Version 7.0.4-7ubuntu1

System	Linux ubuntu-16-lamp 4.4.0-12-generic #28-Ubuntu SMP Wed Mar 9 00:33:55 UTC 2016 x86_64
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.0/apache2
Loaded Configuration File	/etc/php/7.0/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.0/apache2/conf.d
Additional .ini files parsed	/etc/php/7.0/apache2/conf.d/10-mysqlnd.ini, /etc/php/7.0/apache2/conf.d/10-opcache.ini, /etc/php/7.0/apache2/conf.d/10-pdo.ini, /etc/php/7.0/apache2/conf.d/20-calendar.ini, /etc/php/7.0/apache2/conf.d/20-ctype.ini, /etc/php/7.0/apache2/conf.d/20-exif.ini, /etc/php/7.0/apache2/conf.d/20-fileinfo.ini, /etc/php/7.0/apache2/conf.d/20-ftp.ini, /etc/php/7.0/apache2/conf.d/20-gettext.ini, /etc/php/7.0/apache2/conf.d/20-iconv.ini, /etc/php/7.0/apache2/conf.d/20-json.ini, /etc/php/7.0/apache2/conf.d/20-mcrypt.ini, /etc/php/7.0/apache2/conf.d/20-mysqli.ini, /etc/php/7.0/apache2/conf.d/20-pdo_mysql.ini, /etc/php/7.0/apache2/conf.d/20-phar.ini, /etc/php/7.0/apache2/conf.d/20-posix.ini, /etc/php/7.0/apache2/conf.d/20-readline.ini, /etc/php/7.0/apache2/conf.d/20-shmop.ini, /etc/php/7.0/apache2/conf.d/20-sockets.ini, /etc/php/7.0/apache2/conf.d/20-sysvmsg.ini, /etc/php/7.0/apache2/conf.d/20-sysvsem.ini, /etc/php/7.0/apache2/conf.d/20-sysvshm.ini, /etc/php/7.0/apache2/conf.d/20-tokenizer.ini
PHP API	20151012
PHP Extension	20151012
Zend Extension	320151012
Zend Extension Build	API320151012,NTS
PHP Extension Build	API20151012,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	disabled
Zend Memory Manager	enabled
Zend Multibyte Support	disabled
IPv6 Support	enabled
DTrace Support	enabled
Registered PHP Streams	https, ftps, compress.zlib, php, file, glob, data, http, ftp, phar
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, tls, tlsv1.0, tlsv1.1, tlsv1.2
Registered Stream Filters	zlib.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk, convert.iconv.*, mcrypt.*, mdecrypt.*

This program makes use of the Zend Scripting Language Engine:
Zend Engine v3.0.0, Copyright (c) 1998-2016 Zend Technologies
with Zend OPcache v7.0.6-dev, Copyright (c) 1999-2016, by Zend Technologies

Рисунок 2.2

PhpMyAdmin - веб-додаток з відкритим кодом на мові PHP із графічним веб-інтерфейсом для адміністрування бази даних MySQL або MariaDB. phpMyAdmin дозволяє через браузер здійснювати адміністрування сервера MySQL, запускати запити SQL, переглядати та редагувати вміст таблиць баз даних. Ця програма користується великою популярністю у веб-розробників, оскільки дозволяє керувати базу даних MySQL без вводу SQL команд через дружній інтерфейс і з будь-якого комп'ютера під'єднаного до інтернету без необхідності встановлення додаткового програмного забезпечення.

PhpMyAdmin пропонує широкий діапазон документації. Команда phpMyAdmin намагатиметься допомогти, якщо ви зіткнетеся з будь-якою проблемою. Ви можете скористатися різними каналами підтримки, щоб

отримати допомогу. Також phpMyAdmin дуже глибоко задокументовано в книзі, написаній одним із розробників - Mastering phpMyAdmin для ефективного управління MySQL, який доступний англійською та іспанською мовами. Щоб полегшити використання широкого кола людей, phpMyAdmin транслюється на 72 мови та підтримує як LTR, так і RTL мови.

Головні переваги даної системи полягають в тому, що вона проста в використанні і в більшості випадків дозволяє не використовувати команди SQL, тому робота з базою даних є більш простою і підходить для людей які поверхнево знають MySql. Також дана система має велику популярність у веб-програмуванні та є актуальною на сьогоднішній день. PhpMyAdmin може керувати цілим сервером MySQL (потребує супер-користувача), а також єдину базу даних. Для виконання останньої вам буде потрібно належним чином налаштований користувач MySQL, який може читати або записувати лише бажану базу даних. Багато людей ускладнюють розуміння поняття керування користувачами щодо phpMyAdmin. Коли користувач входить до phpMyAdmin, це ім'я користувача та пароль передаються безпосередньо в MySQL. phpMyAdmin не здійснює керування обліковими записами самостійно (крім того, що дозволяють маніпулювати даними про обліковий запис користувача MySQL). Всі користувачі повинні бути дійсними користувачами MySQL. Оскільки інтерфейс phpMyAdmin повністю заснований у нашому браузері, для встановлення файлів phpMyAdmin нам потрібен веб-сервер Apache.

Команда розробника phpMyAdmin робить багато зусиль, щоб зробити phpMyAdmin максимально безпечним. Проте веб-додатки, такі як phpMyAdmin, можуть бути вразливими до ряду атак, і все ще вивчаються нові способи експлуатації. Коли phpMyAdmin показує фрагмент даних користувача, наприклад, щось всередині бази даних користувача, всі спеціальні символи html повинні бути вичерпані. Коли ці зникнення відсутні, шкідливий користувач може заповнити базу даних спеціально зібраним вмістом, щоб змусити іншого користувача цієї бази даних виконувати щось. Це, наприклад, може бути

фрагментом коду JavaScript, який би робив будь-яку кількість неприємних речей.

PhpStorm - інтегроване середовище розробки для PHP. Це інтелектуальне середовище розробки з аналізатором кода, запобігання помилок в коді і автоматизованими засобами рефакторинга для PHP і JavaScript. PhpStorm — розроблений на платформі IntelliJ IDEA, написана на мові програмування Java. Розробники можуть додати функціонал середовища розробки за допомогою встановлення плагінів, розроблених спеціально для платформи IntelliJ, або можна написати свої плагіни.

В ньому спостерігається спеціальна та зручна підсвітка коду. PhpStorm ідеально підходить для роботи з Symfony, Drupal, WordPress, Zend Framework, Laravel, Magento, CakePHP, Yii та іншими фреймворками.

Редактор фактично "отримує" ваш код і глибоко розуміє свою структуру, підтримуючи всі можливості мови PHP для сучасних та застарілих проектів. Вона забезпечує найкраще завершення коду, рефакціонування, попередження помилок на скріптах тощо. Сотні перевірок піклуються про перевірку даного коду під час введення, аналізуючи весь проект. Підтримка PHPDoc, кодовий аранжувальник і форматер, швидкі виправлення та інші функції допоможуть вам написати акуратний код, який легко підтримувати. Рефікціонування свого коду надійно з безпечним перейменування, переміщення, видалення, метод витяжки, вбудована змінна, натискання елементів вгору або витягування елементів вниз, змінити підпис та багато інших рефакторингів. Реакціонування за специфікою мови допоможе вам виконати загальні зміни впродовж декількох кліків, і їх можна безпечно скасувати.

Composer — це менеджер залежностей для PHP. Коли програміст пише свою програму вона так чи інакше залежить від інших (сторонніх) бібліотек як от різні API, фрейморки, ліби що вносять “синтаксичний цукор” і т.д. Так от composer дозволяє правильно керувати цими залежностями: вирішувати конфлікти, автоматично завантажувати правильні простори імен, швидко

стягувати з сервера, оновлювати, шукати і т.д. Всі ці залежності він зберігає в теці з проектом тому це саме менеджер залежностей, а не менеджер пакунків який зберігає залежності в системі.

Composer виконує наступні проблеми:

Ведення проекту зі сторонніми бібліотеками.

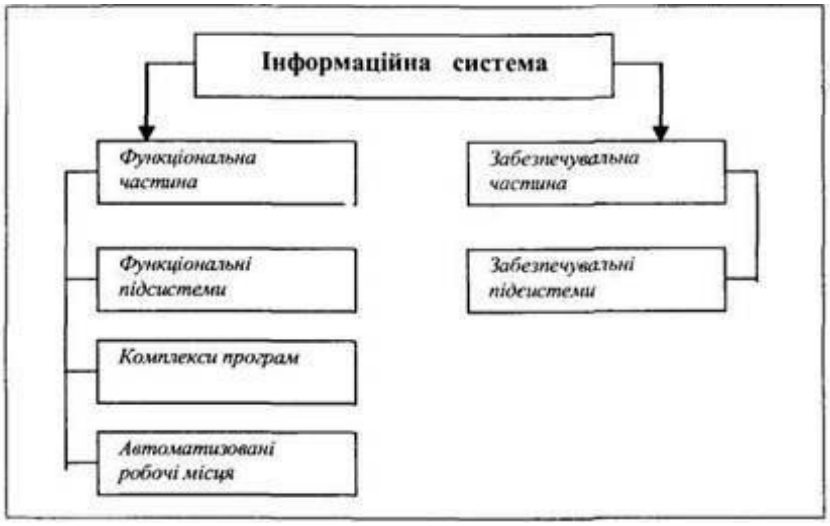
Деякі з ваших бібліотек залежать від інших бібліотек.

Вирішення конфліктів та пріоритети.

Пошук та завантаження в проект потрібних версій бібліотек

2.2 Склад функціональної частини

Функціональна частина — це відносно самостійна частина системи, яку виокремлено за певною ознакою, що відповідає конкретним функціям і завданням управління. Цю підсистему можна розглядати як самостійну систему, що характеризується певним цільовим призначенням, підпорядкованістю, відокремленістю інформаційної бази, методичною спрямованістю обчислень економічних показників і спеціалізацією робіт. Автоматизовану інформаційну систему (АІС) розрізняють на функціональну частину та частину забезпечення, які дальше поділяються на менші елементи підсистеми.



Переважною є частина яка відповідає за фукцію АІС. Пов’язана вона з сервером та об’єктом управління контентом . До неї належать:

1. призначення;
2. функції які виконуються в управліні;
3. функції які оброблять інформацію.

Основними елементами функціональної частини АІС є: блоки, елементи, комплекси задачі, окремі задачі.

Функціональна підсистема – це частично незалежна частина системи, за схожістю функціональних ознак управління.

Функціональна структура АІС має розглядати проблеми інформації кінцевих користувачів, які змінюються в умовах ринку, та відтворювати зміст, а також функцій ті що керують конкретним економічним об'єктом. В АІС має бути гнучка структура і вона має бути відкритою системою, але повина відокремити ті змін у нашу модель та зробити нарощування функціональних частин як потребуєтьсся.

Дана задача виповнюється за поняттям модульності АІС. Кожний прикладний модуль системи виводить інформаційну політику. Головною задачею при розробці даного модулів має орієнтуватися за системою на автоматизовану діяльність об'єкта, а не на впровадження локальних функціональних задач. При цій функції, що робляться, та модулі маютьдивитися з точки потреб наших користувачів, а не програмної реалізації. Комплексність нашої функціональної системи забезпечується для інтеграції модулів в одну систему.

Розробка функціональної частини слід проводити згідно з вимогами висунутими у даних формування задач до системи яка автоматизується до вимог розробки медичної картки пацієнта. Проведемо виділення функцій та виділення функціональних модулів:

1. Управління користувачами:
   1.1. Реєстрація нових користувачів;
   1.2. Видалення користувачів.

2. Управління записами:
   2.1. Створення карток пацієнтів;
   2.2. Створення записів про даних пацієнтів;
3. Сервіс інформування користувачів:
   3.1. Інформування про незаповнені поля;
   3.2. Документація з використання автоматизованої системи.
4. Сервіс обробки:
   4.1. Створення звітності про реєстрацію;

У той же час, як однорівневі, наприклад, так і в дворівневих системах децентралізованої обробки даних функціональними елементами автоматизованих систем є функціонально-спеціалізована ARM (FSAMM). Враховуючи вищезазначене, AWP слід розуміти як сукупність персональних комп'ютерних пристроїв, які за допомогою обчислювальної потужності великого комп'ютера дозволяють користувачеві виконувати служби інформації на робочому місці в обсязі та режимі, необхідних для виконання його функцій.

Загалом, виконання функціональних завдань в АІС може розглядатися як послідовність дій з інформацією, що міститься в базі даних, а також операції вхідних і вихідних даних. Виклик функцій AIS здійснюється за допомогою раціонів. Для кожного користувача система забезпечує індивідуальне харчування тієї, в якій додаються лише необхідні функції, що виключає можливість доступу до функцій системи інших користувачів інформації.

Поліпшення та розвиток функціональної частини АІС відбувається у напрямку розробки та впровадження нових підсистем, засобів масової інформації, завдань. У той же час зміна функціональної структури пов'язана з зміною ринкового середовища, регулювання та ін. Покращення функціональних даних АІС забезпечує повне та ефективне виконання функцій, що виконуються автоматичним методом в а, а також підвищує функціональну придатність АІС, що в кінцевому підсумку відображає покращення управління фінансовими ресурсами держави.

2.3 Склад підсистем забезпечення функціональної частини

2.3.1 Програмне забезпечення

Програмне забезпечення — сукупність програм системи обробки інформації і програмних документів, необхідних для експлуатації цих програм.

PHP — це часто використовувана мова скриптів, розроблена спочатку для веб–програмістів і призначена для створення динамічних сторінок. Абрівеатура PHP походить від (англ) Hepyrtext PreProcesor – процесор гіпертекста. Її можна реалізовувати на більшість веб–серверів, і майже на всіх операційних системах.

Спочатку PHP виступав за «Персональну домашню сторінку», хоча останнім часом він був змінений на «PHP: Hypertext Preprocessor». Однак незалежно від того, PHP є основною частиною будь-якої динамічної веб-сторінки.

Розвиток PHP почався в 1994 році як особистий проект Расмуса Лердорфа, який створив серію Perl-скриптів, які він назвав "Індивідуальними інструментами домашньої сторінки" для підтримки його особистої веб-сторінки. У 1995 році ці інструменти були упаковані та випущені як CGI-файли як "Персональна домашня сторінка / Форми перекладача", яка включала підтримку веб-форм та спілкування з базами даних.

Після того, як він був випущений у світ в цілому, PHP пройшов швидку розробку та розробку, а друга версія PHP / FI була випущена через два роки пізніше в листопаді 1997 року. PHP 3 був випущений в 1998 році з PHP 4 та PHP 5 в 2000 та 2004 роках, відповідно.

PHP 5 - це версія, яка наразі використовується на більшості веб-сайтів, і включає в себе кілька нових функцій, таких як підтримка об'єктно-орієнтованого програмування, послідовний інтерфейс для доступу до бази даних та кілька основних удосконалень.

Хоча PHP залишається в стадії розробки, поточний процес розробки, який в кінцевому підсумку призведе до PHP 6, був повільним, ніж передбачалося

через труднощі додавання підтримки Unicode. У 2010 році було прийнято рішення про перенесення підтримки Unicode до філії, перемістивши всі інші функції, що розробляються, до головного корпусу PHP-коду. Однак у версії 5.4 PHP нарешті додав підтримку Unicode без зміни основної версії.

PHP випущений під ліцензією PHP, яка схожа на загальну публічну ліцензію GNU, за винятком того, що будь-яке похідне програмне забезпечення може не називатися "PHP" і не мати назви "PHP".

В даний час PHP має безліч застосувань, що робить його чудовим інструментом для вирішення будь-якої кількості проектів. Багато основних програмних продуктів, таких як WordPress та phpBB, використовують PHP для виконання завдань, таких як запуск блогу чи форум. PHP також має унікальні можливості, такі як можливість динамічно генерувати зображення в безлічі форматів і доступ до баз даних у багатьох різних форматах.

PHP також має можливість бути вбудованим безпосередньо в веб-сторінку або використовується з командного рядка, що робить його потужним інструментом, який може обробляти що-небудь від відображення інформації, витягнутої з бази даних, до виконання системних завдань запланованим способом.

Важливо пам'ятати про PHP, що це попередній процесор, а це означає, що будь-які скрипти PHP на веб-сторінці виконуються перед тим, як сторінка відображається. Це означає, що будь-які скрипти PHP на сторінці не можуть змінити цю сторінку після її відображення. Існує кілька способів подолання цього обмеження, наприклад AJAX (Asynchronous Java Script and XML), що дозволить вам змінювати те, що знаходиться на сторінці, не оновлюючи всю сторінку, але ці технології та методи не підпадають під дію цієї статті. .

Найпоширенішим використанням PHP є доступ до бази даних, аналіз результатів з цієї бази даних та відображення результатів на веб-сторінці. Ось чому PHP - остання частина загального абревіатуру "LAMP", що означає "Linux, Apache, MySQL і PHP". Встановлення LAMP - це одна з найпоширеніших конфігурацій веб-сервера і поєднує в собі потужний веб-

сервер Apache з PHP та MySQL, що дозволяє створювати вражаючі надійні веб-сторінки та керувати даними. Фактично ці інструменти часто настроюються, щоб працювати разом з малою або без додаткової конфігурації.

Важливо відзначити, що кожен, хто регулярно використовує PHP, полягає в тому, що PHP найчастіше виконує з тими самими правами, що і веб-серверне програмне забезпечення. З точки зору безпеки важливо пам'ятати, що якщо щось знаходиться у вашому веб-каталозі на вашому сервері, неправильно написаний скрипт PHP може мати доступ до нього.

На закінчення, PHP є потужною мовою, яка стала однією з рушійних сил Інтернету, і кожен, хто розглядає кар'єру в веб-розробці, повинен зробити навчання PHP першочерговим.

2.3.2 Апаратне забезпечення

Апаратне забезпечення (англ. hardware) — комплекс технічних засобів, який включає ЕОМ: зовнішні пристрої, термінали, абонентські пункти тощо, які необхідні для функціонування тієї чи іншої системи; фізична частина ЕОМ.

Згідно обраних технологій, архітектури, платформи, структури функціональної частини та програмного забезпечення до апаратного забезпечення висуваються такі вимоги:

–PHP>= 7.0.0;

–OpenSSL PHP Extension;

–PDO PHP Exstension;

–Mbstring PHP Exstension;

–Tokenizer PHO Exstension;

–XML PHP Exstension;

Якщо ми встановили PHP локально, і хочемо використовувати вбудований сервер PHP для роботи з нашою програмою, ми можемо використовувати команду serve artisan.

Багато нових вбудованих системних плат з високопродуктивними обчислювальними можливостями стали нормою в світі Інтернету. Незалежно від того, чи використовуєте ви Малину Пі, Beaglebone чи багатьох інших, незвичайно знайти вбудоване в мережу устаткування, здатне підтримувати функціональність обласного або простого сервера.

Багато хто з цих плат, як можна було б очікувати, можуть інтерфейсувати через цифрові або аналогові штифти до міріаду датчиків і виконавчих механізмів. Але вони також мають роз'єми RJ45, з'єднані з контролерами Ethernet. І їх операційна система здатна розміщувати серверне програмне забезпечення (наприклад, Apache), яке включає PHP.

Метод Extract витягує потрібний набір точок і обробляє повний цикл доступу через виконуваний файл. Фактично, виконуваний файл обробляє як інтерфейс сенсора, так і деталі переносу вихідних даних. У свою чергу, клас Sensor може використовуватися для виведення даних відформатованого датчика на мережевий запит і, необов'язково, для виконання додаткової обробки.

Одна з великих переваг програмного забезпечення Open Source полягає в тому, що вона забезпечує можливість адаптації до нових середовищ. Це стосується PHP. Незважаючи на те, що спочатку він був призначений як модуль для веб-сервера Apache, PHP з тих пір прийняв стандарт ISAPI, що дозволяє йому працювати однаково добре з інформаційним сервером Microsoft. Що стосується вимог до апаратних засобів, ми особисто бачили, як працює PHP на 100-МГц Pentium-машинах під управлінням Slackware Linux та Windows NT відповідно. Виконання було добре для використання в якості особистого середовища розробки. Звичайно, сайт, який очікував отримати тисячі запитів на день, потребує швидшого обладнання. Незважаючи на те, що для порівняння PHP-сайту з плоским HTML-сайтом потрібні додаткові ресурси, вимоги не різко відрізняються. Незважаючи на мій приклад, ви не обмежуєтеся обладнанням Intel. PHP працює однаково добре на процесорах PowerPC та Sparc.

Вибравши операційну систему, ви маєте загальний вибір між Windows і UNIX-подібною операційною системою. PHP буде працювати на Windows 95 і 98, хоча ці операційні системи не підходять для великих трафічних веб-серверів. Він також буде працювати в Windows NT та його наступнику Windows 2000. Для операційних систем UNIX, PHP добре працює з Linux і Solaris, а також іншими. Якщо ви вибрали систему на основі PPC, наприклад Macintosh, ви можете вибрати LinuxPPC, версію Linux. Ви можете переслідувати комерційний WebTen веб-сервер, який працює в ОС Macintosh. Чад Каннінгем запропонував патчі для складання PHP в Apple OS X. У 1999 році Брайан Хавард додав підтримку IBM OS / 2.

PHP все ще найкраще працює з веб-сервером Apache. Але тепер він дуже добре працює з IIS. Він також складається як модуль для веб-сервера fhttpd. Ви можете виконувати роботу PHP практично з будь-яким веб-сервером за допомогою версії CGI, але я не рекомендую цю настройку для веб-сайтів виробництва. Якщо ви використовуєте UNIX, ми рекомендуємо скомпілювати PHP як модуль Apache. Якщо ви використовуєте Windows NT, перейдіть до IIS.

Якщо використовувати Linux, ми можете легко знайти RPM для Apache та PHP, але ця установка може не включати будь-яку бажану функцію PHP. Я рекомендую цей маршрут як дуже швидкий старт. Ви завжди можете перейти до складання Apache та PHP з нуля пізніше. PHP буде компілюватися на більшості версій операційних систем UNIX, включаючи Solaris і Linux. Якщо ви коли-небудь склали програмне забезпечення, яке ви знайшли в Інтернеті, у вас буде трохи проблем з цією установкою. Якщо у немає досвіду вилучення файлів з tar архіву та виконання файлів make, ви можете покластися на ваш системний адміністратор або хтось інший більш досвідчений. Вам потрібно буде мати кореневі права для повного встановлення PHP.

Сценарії PHP - це лише текстові файли, ми можемо редагувати і створювати їх, як і HTML-файли. Звичайно, ми можемо створювати файли з блокнотом і використовувати ftp для їх завантаження по черзі. Але це не

ідеальні переживання. Одна зручна функція нових редакторів - це вбудований FTP. Ці редактори можуть відкривати файли на віддаленому веб-сервері як на локальному диску. Один клік зберігає їх на віддаленому веб-сервері. Інша особливість, якою ви можете насолоджуватися - виділення синтаксису. Це призводить до того, що ключові слова PHP будуть забарвлені, щоб допомогти вам швидше прочитати код.

Отже, як бачимо для роботи розроблюваного веб-додатку нам потрібні найкращі апаратні потужності, що дозволяє використовувати його на багатьох хостингах.

2.3.2 Інші види забезпеченння

До інших видів забезпечення відноситься організаційне забезпечення, правове, лінгвістичне та ергономічне.

Організаційну підтримку слід розуміти як узгодження місця, часу та мети спільної роботи окремих виконавців, команд та технічних засобів. Вона повинна бути запроваджена та керована певними правилами взаємодії, які формують правовий та моральний кодекс і формують правову базу. Те, що організаційна підтримка, базується на нормативно-правових актах правової охорони, і юридичне положення втілюється в організаційне забезпечення.

Організаційна підтримка інформаційної системи включає набір інструментів, методів та відповідного персоналу. Вона повинна забезпечити:

- проведення техніко-економічного обґрунтування існуючої системи управління, відбору та встановлення завдань побудови інформаційної системи на етапі розробки та реалізації;

- регулювання взаємодії персоналу з ансамблем технічними засобами та між собою в процесі вирішення контрольних завдань, моніторингу ефективності системи управління на етапі функціонування інформаційної системи.

На етапі проектування організаційна підтримка виконує наступні завдання:

- аналіз існуючих систем управління та формулювання напрямів підвищення їх ефективності;

- вибір та виклад завдань управління;

- формулювання вимог до ансамблю технічних засобів;

- розробка організаційних рішень щодо складу, структури, організації та методології вирішення проблем управління в інформаційній системі, складу робочих процедур та пояснення їх реалізації.

На етапі функціонування інформаційної системи організаційна підтримка вирішує такі завдання:

- впровадження методів управлінських завдань;

- організація персоналу та ансамблю технічних засобів інформаційної системи;

- контроль та аналіз ефективності управління;

- формування пропозицій щодо вдосконалення та розвитку інформаційної системи.

Основою ефективного розвитку підприємств є регуляторна база, що базується на сукупності законів, необхідних для регулювання та забезпечення. Юридична підтримка - це сукупність загальновизнаних заходів, виражених у нормативних актах, які встановлюють та встановлюють організацію інформаційної системи, її мету, завдання, структуру та функції (правовий статус інформаційної системи та її підрозділів), призначені для регулювання створення та функціонування інформаційної системи. Юридична безпека базується на правовому підході, який розглядає соціальне управління як організацію в чомусь, включаючи виконавчу та адміністративну діяльність державних органів, спрямовані на реалізацію законів та інших нормативних актів, прийнятих владою та керівництвом. Юридичний підхід аналізує місце та роль закону в управлінні, визначає зміст законної виконавчої та регуляторної

діяльності органів державної влади та органів управління, робить рекомендації щодо його вдосконалення.

Функції структурно-правового управління виконуються у формі нормативно-правових актів, планів, регламентів і способів, обов'язкові для всіх гостей, які визначають різні матеріали у сферах планування та управління. Нормативно-правові акти - це конфігурація виразу та встановлення правових загально визнаних заходів, сукупність яких формує правове Основи управління, яке є ієрархічним. Нормативно-правові акти поділяються на законодавчі, нормативні акти міністерств та відомств, акти місцевих органів влади та управління, місцеві акти, окремі акти.

На етапі функціонування інформаційної системи юридична підтримка включає:

- статус інформаційної системи у конкретних сферах державного управління;
- правова позиція на тему компетенції зв'язків інформаційної системи та організації їх діяльності;
- права, обов'язки та відповідальність персоналу інформаційної системи;
- правовий статус певних видів управлінських процесів в інформаційній системі;
- порядок отримання та використання інформації в інформаційній системі, процедури його збору, реєстрації, зберігання, передачі та обробки;
- порядок отримання та використання ансамблю технічних засобів, програмного забезпечення, інформації та інших видів підтримки.

Лінгвістичне забезпечення охоплює сукупність науково-технічних термінів ти інших мовних засобів, правил формалізації мов, методів стиску запису інформації, засобів діалогу людини і обчислювальної системи. Лінгвістичне забезпечення включає в себе:

- інформаційні мови для опису структурних одиниць баз даних інформаційної системи (документів, показників, реквізитів);

- мови управління, маніпулювання і обміну даними в банку даних інформаційної системи;

- мовні засоби інформаційно-пошукових систем;

- мовні засоби системи автоматизованого проектування;

- діалогові мови;

- словники термінів і визначень.

Ергономічне забезпечення охоплює сукупність методів і засобів, призначених для створення оптимальних умов для ефективної діяльності і навчання операторів з складу персоналу інформаційної системи. Ергономічне забезпечення включає в себе:

- комплекс документації, яка містить ергономічні вимоги до робочих місць і здійснює експертизу робочих місць;

- комплекс методів, учбово-методичних матеріалів і технічних засобів підготовки персоналу до роботи;

- комплекс методів і засобів, які забезпечують професійний відбір.

В плані ергономічного забезпечення на етапах проектування інформаційної системи визначається ступінь і рівень участі людини в системі управління, вимоги до форми представлення інформації, умови оточуючого середовища діяльності людини, порядок роботи і відпочинку персоналу, нормативи навантаження і надійності персоналу; вимоги до технічних засобів, способи взаємодії персоналу і технічних засобів.

В реальних інформаційних системах загальне число видів забезпечення, що підтримують достатнє та повне функціонування організації може бути і меншим. Обов’язковими складовими інформаційної системи є лише підсистеми програмного та апаратного забезпечення. Функції інших видів забезпечення менш значимі і можуть об’єднуватися і групуватися або входити в основні підсистеми.

3. РЕАЛІЗАЦІЯ ПРОТОТИПУ ВЕБ-ОРІЄНТОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ «ЕЛЕКТРОННА МЕДИЧНА КАРТКА ПАЦІЄНТА»

3.1 Структура та особливості реалізації інформаційного забезпечення

Інформаційне забезпечення – це сукупність проектних рішень за розмірами, розміщенням, формами організації інформації, яка циркулює в автоматизованій системі.

Інформаційне забезпечення в значній мірі визначає інтелект системи, оскільки містить всю використовувану інформацію, оперує нею і здійснює інформаційний обмін всередині і зовні інформаційної системи. До інформаційного забезпечення висуваються серйозні вимоги. Інформація повинна бути достовірною, своєчасною, постійно обновлюваною, представленою в зручній для користувача формі, доступною і повною. Неповність інформації часто є причиною прийняття нераціональних і несвоєчасних управлінських рішень.

Інформаційне забезпечення містить не тільки різноманітні дані, але і систему доступу до них, яка реалізується програмними засобами. В цьому проявляється тісний взаємозв’язок інформаційного і програмного забезпечення, яке надає ці засоби.

На основі отриманих результатів в попередніх розділах, був створений прототип автоматизованого рішення, розроблений за допомою фреймворку Laravel, та бази даних для збереження інформації, створеної на мові MySQL.

Для подальшої роботи над прототипом додатку необхідно встановити стек LAMP. LAMP – це абревіатура вільного програмного забезпечення з відкритим кодом, до якого входять операційна система Linux, веб-сервер Apache, система керування базами даних MySQL, та інтерпретатор Perl/PHP/Python – основні компоненти для побудови життєздатного багатоцільового веб-сервера.

Для початку необхідно встановити веб-сервер Apache. Для цього достатньо скористатися менеджером пакетів Ubuntu apt, використовуючи команди, які зображені на рисунку 3.1.

```
$ sudo apt-get update
$ sudo apt-get install apache2
```

Рисунок 3.1 – Встановлення Apache

Далі потрібно встановити MySQL. Для того щоб завантажити та встановити програмне забезпечення достатньо скористатися аналогічним способом, але використавши команду, яку можна побачити на рисунку 3.2.

```
$ sudo apt-get install mysql-server
```

Рисунок 3.2 – Встановлення MySQL

Наступним кроком буде встановлення PHP, а також додаткових пакетів, за допомогою яких PHP зможе працювати з сервером Apache та спілкуватися з MySQL. Для встановлення даних компонентів скористаємося командою, кА зображена на рисунку 3.3.

```
$ sudo apt-get install php libapache2-mod-php php-mcrypt php-mysql
```

Рисунок 3.3 – Встановлення PHP та допоміжних пакетів

Для роботи з MySQL потрібно встановити веб-додаток phpMyAdmin (рисунок 3.4).

```
sudo apt-get install phpmyadmin apache2-utils
```

Рисунок 3.4 – Встановлення додатку phpMyAdmin

Далі необхідно встановити Composer (Рисунок 3.5).

```
$ mv composer.phar /usr/local/bin/composer
$ chmod +x /usr/local/bin/composer
```

Рисунок 3.5 – Встановлення Composer

Наступним етапом буде встановлення самого фреймворку Laravel. Він використовую Composer для управління залежностями, для цього достатньо скористатися командою create-project (рисунок 3.6).

```
composer create-project laravel/laravel {directory} "5.0.*" --prefer-dist
```

Рисунок 3.6 – Встановлення Laravel

Після встановлення фреймворку необхідно створити базу даних в якій будуть зберігатися дані про пацієнтів. Створення бази даних відбувається в phpMyAdmin (рисунок 3.7).

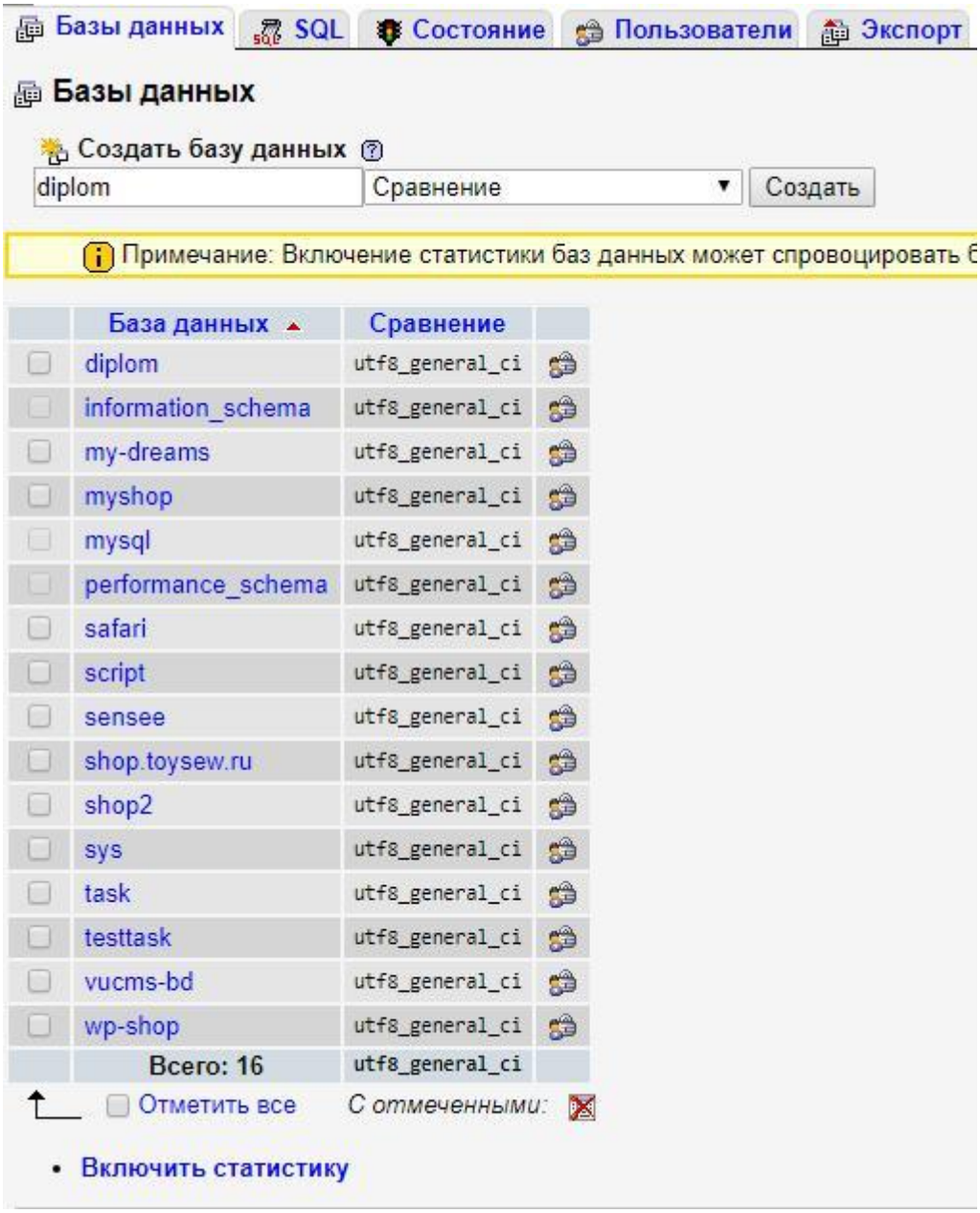


Рисунок 3.7 – Створення бази даних в phpMyAdmin

Для того щоб створити таблиці в базі даних можна скористатися системою об'єктно-реляційного відображення Eloquent ORM – красива і проста реалізація ActiveRecord в Laravel для роботи з базами даних. Кожна таблиця має відповідний клас-модель, який використовується для роботи з цією таблицею. Для початку створимо таблицю для користувачів прототипу електронної медичної картки, яка буде мати назву users (лістинг 3.1).

Лістинг 3.1 – Приклад створення таблиці users

```
public function up()
{
   Schema::create('users', function (Blueprint $table) {
       $table->increments('id');
       $table->string('name');
       $table->string('email')->unique();
       $table->string('password');
       $table->rememberToken();
       $table->timestamps();
   });
}
```

Структура даної таблиці зображена в таблиці 3.1. Вона включає дані щодо імені та електронної адреси користувача.

Таблиця 3.1 – Структура та опис таблиці користувачів додатку

Назва атрибуту	Тип даних	Короткий опис
_ID	Integer	Первинний ключ, унікальний ідентифікатор рядків
name	String	Ім’я користувача
email	String	Емейл користувача
password	String	Пароль
remember_token	String	Токен
created_at	Timestamps	Дата створення
updated_at	Timestamps	Дата оновлення

Таблиця профіля забезпечує перхід до кабінета користувача, за його ідентифікатором.

Лістинг 3.2 – Приклад створення таблиці profiles

```
public function up()
{
   Schema::create('profiles', function (Blueprint $table) {
       $table->increments('id');
```

```
        $table->integer('user_id');
        $table->timestamps();
    });
}
```

Дана таблиця містить ідентифікатор користувача, та відомості про створення і оновлення даної таблиці. Детально в таблиці 3.2.

Таблиця 3.2 – Структура та опис таблиці профіль прототипу

Назва атрибуту	Тип даних	Короткий опис
_ID	Integer	Первинний ключ, унікальний ідентифікатор рядків
user_id	Intager	Ідентифікатор користувача
created_at	Timestamps	Дата створення
updated_at	Timestamps	Дата оновлення

Великими перевагами структури бази даних є відсутність надмірності даних. Для цього потрібно розподілити інформацію з кількох окремих тематично організованих таблиць, щоб кожен факт був представлений один раз. Таблиця профілів та таблиця користувачів має зв'язок один до одного.

Лістинг 3.3 – Приклад зв’язку таблиці користувачів до профіля

```
public function profile()
{
    return $this->hasOne(Profile::class, 'user_id', 'id');
}
```

Лістинг 3.4 – Приклад зв’язку таблиці профіля до користувачів

```
public function user()
{
    return $this->belongsTo(User::class, 'user_id');
}
```

Так як таблиці для користувачів вже створені, потрібно створити таблицю в якій будуть зберігатися дані про конкретних пацієнтів (таблиця 3.3)

Таблиця 3.3 – Структура та опис таблиці data_users прототипу

Назва атрибуту	Тип даних	Короткий опис
_ID	Integer	Первинний ключ, унікальний ідентифікатор рядків
full_name	String	Прізвище, ім’я, по батькові
sex	String	Стать
phone	String	Телефон
home	String	Місце проживання
work_place	String	Місце роботи
position	String	Посада
dispensary	String	Приналежність чи ні пацієнта до диспансерної групи
preferential_categories	String	Контингент пільгових категорій
preferential_categories_num	String	Номер пільгового посвідчення
disease	String	Захворювання
date	String	Дата взяття на облік, або зняття
user_id	Intager	Ідентифікатор користувача
created_at	Timestamps	Дата створення
updated_at	Timestamps	Дата оновлення

3.2 Структура та особливості реалізації алгоритмічного забезпечення

Алгоритм – це послідовність, система, набір систематизованих правил виконання обчислювального процесу, що обов'язково приводить до розв'язання

певного класу задач після скінченного числа операцій. При написанні комп'ютерних програм алгоритм описує логічну послідовність операцій. Для візуального зображення алгоритмів часто використовують блок-схеми. Кожен алгоритм є списком добре визначених інструкцій для розв'язання задачі. Починаючи з початкового стану, інструкції алгоритму описують процес обчислення, які відбуваються через послідовність станів, які, зрештою, завершуються кінцевим станом. Перехід з одного стану до наступного не обов'язково детермінований — деякі алгоритми містять елементи випадковості. Для створення програми по автоматизації реєстрації пацієнтів була використана не процедурна мова програмування SQL.

Алгоритм роботи системи з працівником медичного закладу наведений на рисунку 3.8.

Рисунок 3.8 - Алгоритм роботи системи з працівником медичного закладу

3.3 Інтерфейс користувача та інструкція по використанню Розроблювану систему можна поділити на серверну та клієнтську частини. Відповідно до даного розподілу будемо описувати інструкцію по використанню

та інтерфейс користувача. Перед тим, як увійти до системи, необхідно пройти реєстрацію. Вікно реєстрації зображено нижче. У поле «Name» вводиться ім’я працівника, у поле E-Mail Address вводиться адрес електронної пошти, відповідно в поле Password вводиться пароль, а в поле Confirm Password – підтвердження паролю.

Рисунок 3.9 – Голова сторінка

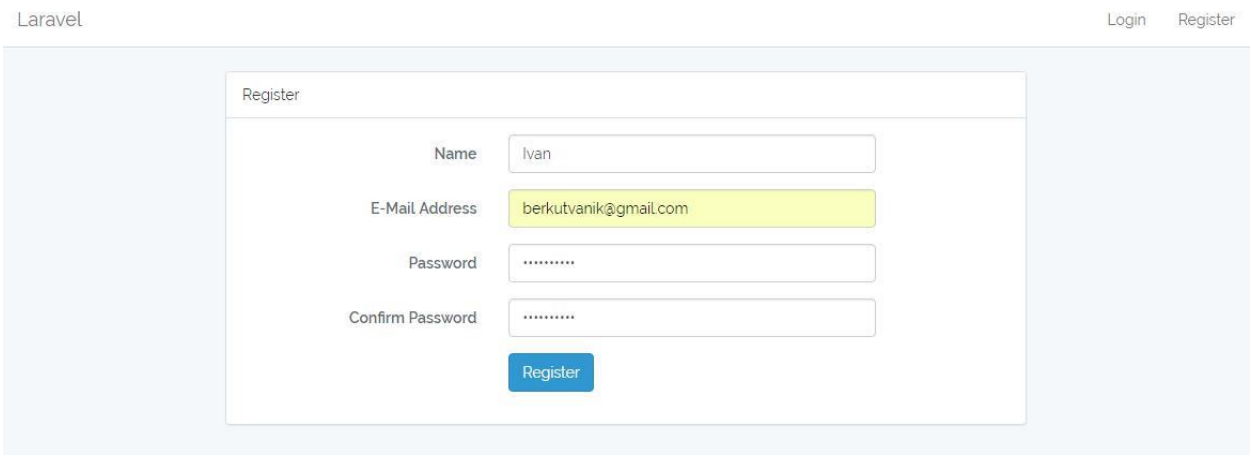


Рисунок 3.10 – Вікно реєстрації працівника медичного закладу

У випадку введення існуючої в базі даних електронної адреси, або невірного підтвердження паролю користувач отримає помилку та не буде зареєстрований (рисунок 3.11 – 3.12).

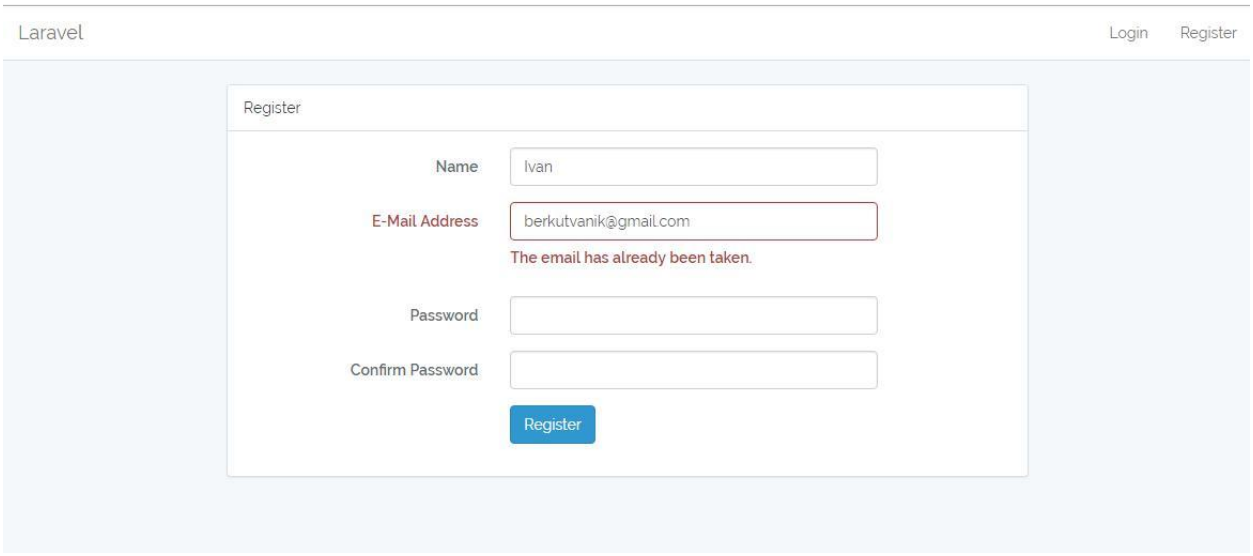


Рисунок 3.11 – Введення існуючої адреси електронної пошти

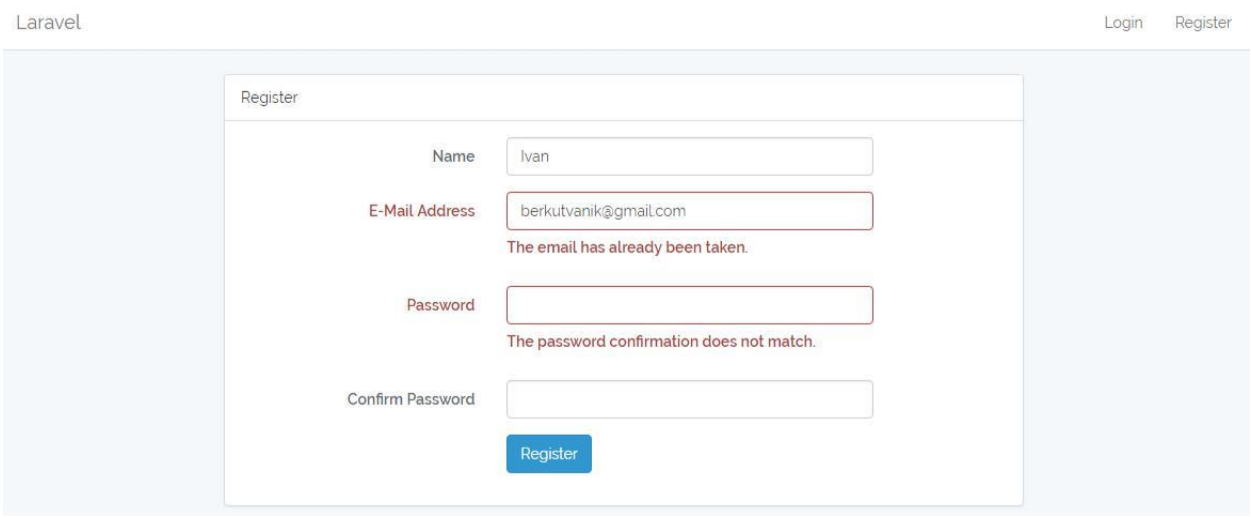


Рисунок 3.12 – Невірне підтвердження паролю

Після успішної реєстрації користувач потрапляє на сторінку профілю де вказана інформація відносно заповнення картки, та відображено головне меню (рисунок 3.13). До головного меню входять такі пункти як «Профіль», «Зареєструвати пацієнта», «Пацієнти», та пункт авторизованого користувача, на тиснувши на який можна вийти із системи.

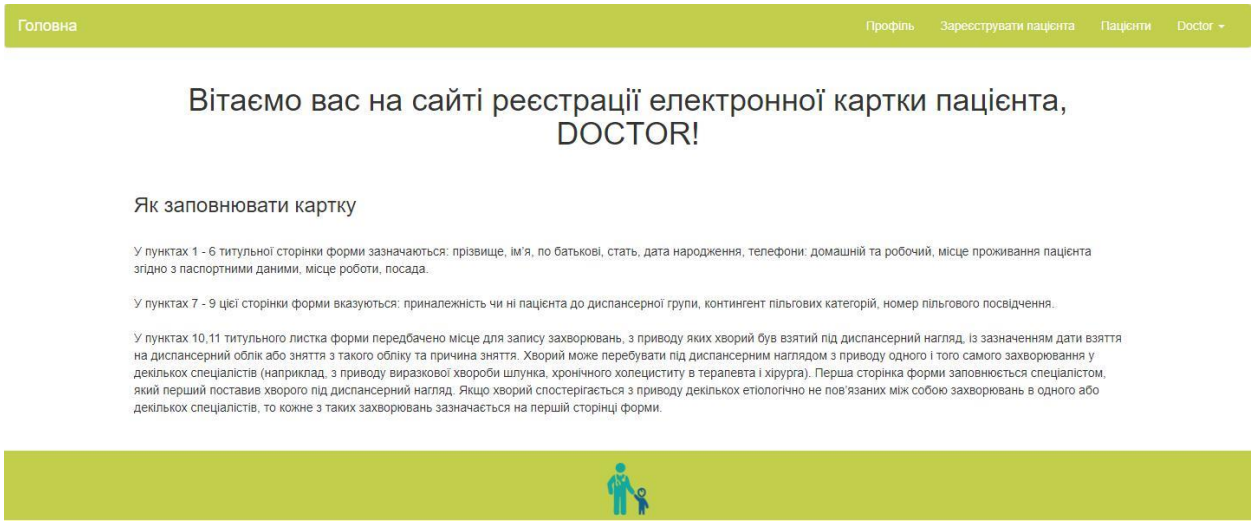


Рисунок 3.13 – Сторінка профілю

При на тисканні на пункт «Зареєструвати пацієнта», користувач потрапляє на сторінку реєстрації пацієнта (рисунок 3.14), де він має можливість вказати його дані, а саме:

- Прізвище, ім’я, по-батькові;
- Стать;
- Телефон;
- Місце проживання;
- Місце роботи;
- Посаду;
- Приналежність чи ні пацієнта до диспансерної групи;
- Контингент пільгових категорій;
- Номер пільгового посвідчення;
- Захворювання;
- Дату взяття на облік, чи зняття.

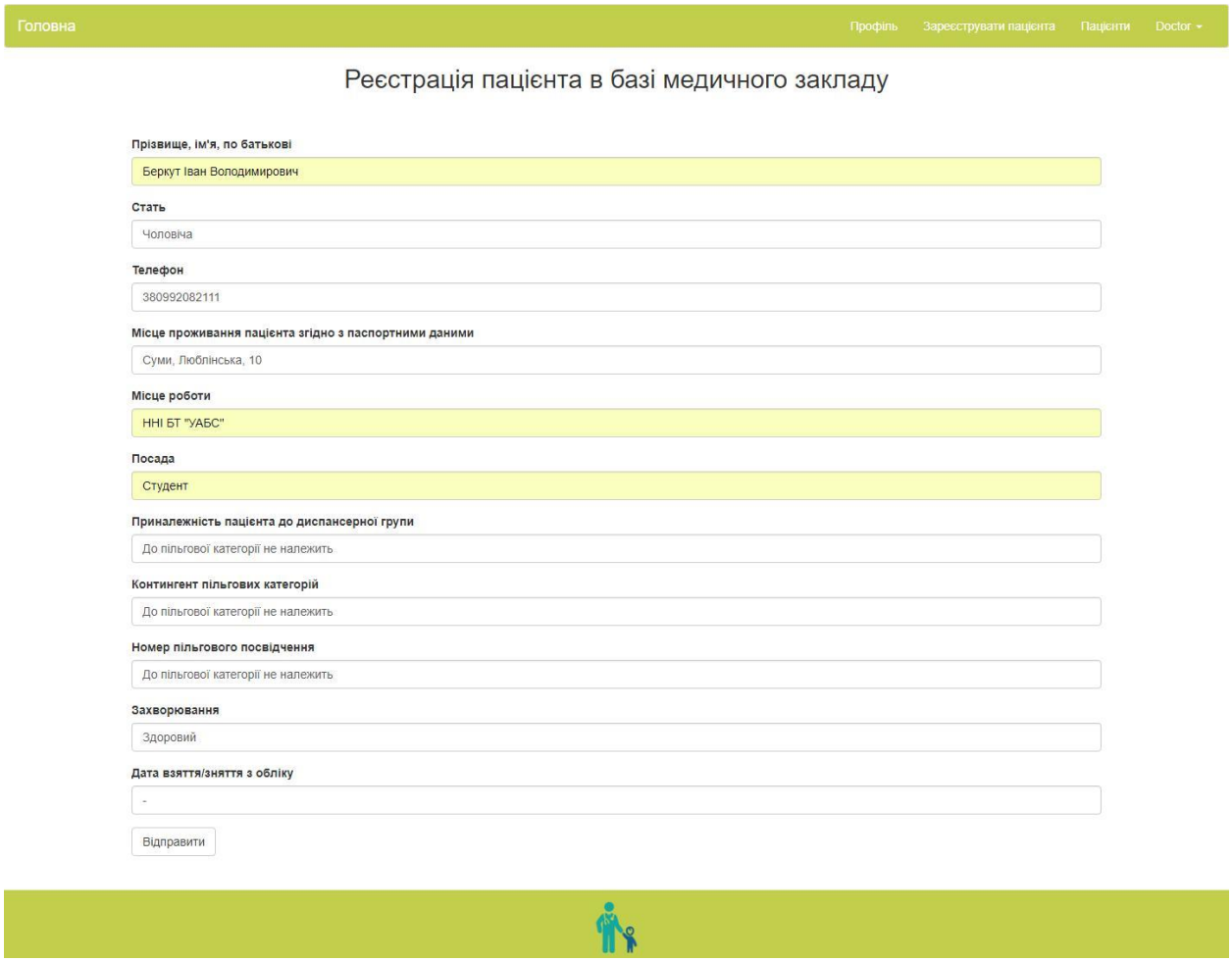


Рисунок 3.14 – Сторінка реєстрації пацієнта

Після натискання на кнопку «Відправити» користувач отримує повідомлення про успішну реєстрацію пацієнта (рисунок 3.15), дані заносяться до бази даних.

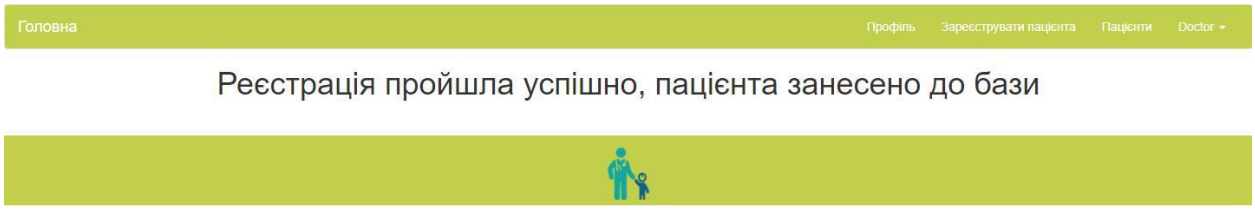


Рисунок 3.15 – Успішна реєстрація пацієнта

Для того щоб переглянути, редагувати або видалити дані пацієнта, потрібно спочатку перейти на сторінку «Пацієнти», посилання на яку знаходиться в головному меню. Сторінка «Пацієнти» зображена на рисунку 3.16.

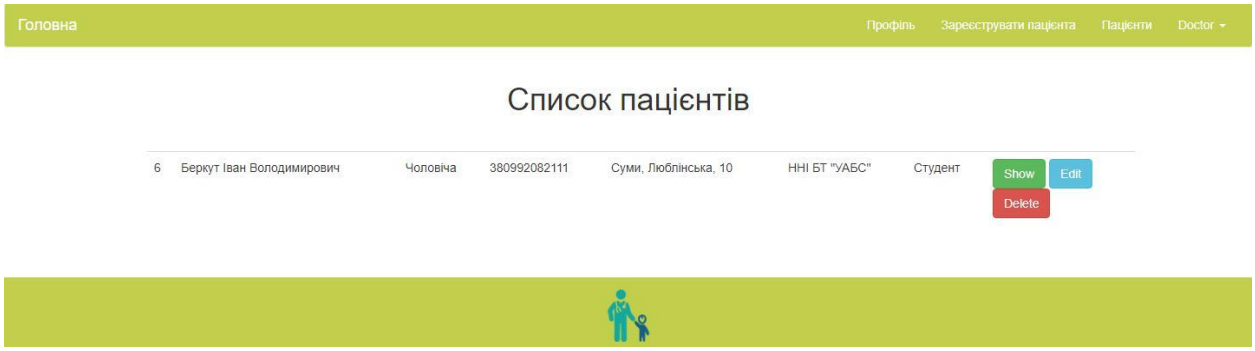


Рисунок 3.16 – Сторінка «Пацієнти»

3.4 Оцінка очікуваних ефектів від впровадження веб-орієнтованої інформаційної системи

Використання автоматизованої системи реєстрації пацієнтів до медичного закладу дає змогу вирішити низку проблем, таких як швидкість обслуговування в медичних уставовах, якість обслуговування, ефективність та інші.

Як відомо продуктивність та ефективність від провадження автоматизованої системи визначають порівнянням результатів її роботи і затрат всіх видів ресурсів, необхідних для її створення і розвитку.

Вкладення ресурсів у створення програмного продукту розраховуються за формулою:

$$K \quad K_1 \quad K_2 \quad K_3 , \qquad (3.1)$$

де K_1 – витрати на устаткування, грн.;

K_2 – витрати на ліцензійні програмні продукти, грн.;

K_3 – витрати на створення програмного продукту, грн.

Приймаємо К1 = 0, оскільки всі вищі учбові заклади використовують комп’ютери для своєї роботи, К2 = 0, оскільки не передбачено закупівлю додаткових програмних продуктів для впровадження системи.

Витрати на створення програмного продукту $K_З$ розраховуємо по формулі:

$$K_З = З_1 + З_2 + З_3, \quad (3.2)$$

де $З_1$ – витрати праці програмістів-розробників, грн.;

$З_2$ – витрати комп'ютерного часу, грн.;

$З_3$ – непрямі (накладні) витрати, грн.

Витрати праці програмістів-розробників:

$$З_1 = \sum_{k=1}^{K} N_k \cdot r_k \cdot T_k \cdot K_{зар}, \quad (3.3)$$

де N_k – кількість розробників к-й професії, чол.;

r_k – годинна зарплати розробника к-й професії, грн.;

T_k – трудомісткість розробки для к-го розробника (кількість витраченого розробником часу), ч.;

Кзар = 1,3685– коефіцієнт відрахувань до фонду заробітної плати. Візьмемо, що N_k = 1, оскільки в розробці програмного продукту приймала участь лише 1 людина.

Годинну заробітну плату працівника розрахуємо за формулою:

$$r_k = M_k / F_{кмес}, \quad (3.4)$$

де M_k – місячна зарплата к-го розробника, грн.;

Fкмес – місячний фонд часу його роботи, година.

Приймаємо місячну заробітну плату програміста-розробника по Сумській області рівну 4000 грн., тоді маємо, що кількість календарних днів на 2017 рік = 365, кількість святкових днів = 10, кількість вихідних днів = 104, кількість днів, робота в які не проводиться = 114, кількість робочих днів = 251, кількість днів, що передують святковим, в які робочий день коротший на 1 годину в день = 4. Всього робочих годин за рік, при 40 годинному робочому тижні = 2004 години. Всього робочих годин за місяць = 2004/12=167 год.

Тоді, годинна зарплата розробника, враховуючи це, буде обчислена як:

$r_k = \frac{4000}{167}$ 24,39 грн./год.

Трудомісткість розробки $T_к$ включає час виконання робіт і в даному випадку є рівним 300 годин.

Тепер розрахуємо витрати праці програміста-розробника: $З_1$=1·24,39·300·1,3685 = 10013,32 грн.

Витрати комп'ютерного часу розрахуємо за формулою:

$$З_{2} = C_k \cdot F_0, \tag{3.5}$$

де C_k – вартість комп'ютерної години, грн.;

F0 – витрати комп'ютерного часу на розробку програми, годин.

Вартість комп'ютерної години обчислюється по формулі:

$$C_k = C_A + C_Э + C_{ТО}, \tag{3.6}$$

де C_A – амортизаційні відрахування, грн.;

$C_ф$ – енерговитрати, грн.;

$C_{ТО}$ – витрати на техобслуговування, грн.

Амортизаційні відрахування знайдемо за формулою:

$$C_A = C_i N_A / F_{год}, \quad (3.7)$$

де C_i = 6000 грн. – балансова вартість і-го устаткування, яке використовувалося для створення , грн.;

N_A – річна норма амортизації і-го устаткування, долі;

$F_{год}$ – річний фонд часу роботи і-го устаткування, година

Відповідно до чинного законодавства квартальна норма амортизації основних фондів 4 групи, які були задіяні у розробці складає 15%, тоді річна норма амортизації буде дорівнювати $N_4 = 0,6$.

$F_{год} = 2004$ год.

$C_A = 6000 \cdot 0,6/2004 = 1,8$ грн.

Енерговитрати розрахуємо за формулою:

$$C_Э = P_Э C_{кВт}. \quad (3.8)$$

Комп'ютер сучасної моделі в середньому затрачає 800 Вт за годину, тоді $P_e = 0,08$ кВт/год.

Вартість 1 кВт/год. для споживачів другого класу (не промислові підприємства) становлять $C_{кВт} = 93,46$коп. за кВт/год.

$C_Э = 0,08 \cdot 0,9346 = 0,0747$ грн./год.

Витрати на техобслуговування розраховуємо за формулою:

$$C_{ТО} = r_{ТО}, \quad (3.9)$$

де $r_{ТО}$ – годинна зарплата працівника обслуговуючого грн; устаткування,

приймаємо $r_{ТО} = 3000/167 = 17,96$ грн. /година;

– періодичність обслуговування:

$$N_{TO} / F_{mec}\ , \qquad (3.10)$$

де N_{TO} – кількість обслуговувань устаткування в місяць, приймаємо $N_{то} 1$

Місячний фонд часу роботи устаткування 167 годин.

= 1/167=0,006.

Витрати на техобслуговування складуть: C_{TO} =17,97·0,006 = 0,11 грн.

Звідси вартість комп'ютерної години: C_k = 1,8 + 0,0744 + 0,11 = 1,98 грн.

Отже витрати комп'ютерного часу складуть: $З_2$ = 1,98·300 = 594 грн.

Непрямі витрати приймемо: $З_З$ = 400 грн..

Звідси $К_З$ = 400 + 594 + 10013,32 = 11007,32 грн.

Витрати на створення системи складають: $К$ = 0 + 0 + 11007,32 = 11007,32 грн.

Тепер можна розрахувати річний ефект від впровадження автоматизованої системи та термін окупності системи.

Річна економія від зниження витрат на розробку проекту визначається за формулою:

$$Ер = Едв – Епв\ , \qquad (3.11)$$

де $Ер$ – сума річної економії, грн.;

$Едв$ – річні витрати на обробку інформації до впровадження системи, грн.;

$Епв$ – річні витрати на обробку інформації після впровадження системи, грн.

Нехай середні витрати часу на ведення обліку та аналізу стану реєстрації субсидії та пільгових надбавок складають 30 хв.. З допомогою даної програми дані витрати можна скоротити до 15 хв. Тоді можна розрахувати економію часу, яка дорівнює 50%. За день здійснюється декілька подібних перевірок.

Середнє вивільнення часу за один робочий день складатиме 30 хв., або 12,5 % робочого дня.

Користувач системи, нехай, в середньому отримує заробітну плату у розмірі 5000 грн. в місяць. Тоді фонд оплати праці одного працівника складає в середньому 60000.

Звідси річна економія складатиме, згідно до формули:

E_p = (60000– 60000*0,125) = 7500 грн./рік.

Розрахуємо термін окупності капіталовкладень – період часу, протягом якого окупаються витрати на автоматизовану систему визначаються по формулі:

$$T_p = K / Ep. \qquad (3.12)$$

T_p = 11007,32 / 7500 = 1,47 років.

При ефективному використанні капіталовкладень розрахунковий термін окупності T_p повинен бути менше нормативного $T_н$ 2, 4, у нашому випадку $T_p < T_н$. так як 0,85 < 2,4.

Отже, як бачимо розробка і використання програмного продукту є економічно доцільною, так як річна економія складе 7500 грн. за рік, а термін окупності капіталовкладень складає 1,47, тобто проект окупиться за півтора року.

ВИСНОВКИ

Дана дипломна робота присвячена розробці прототипу веб-орієнтованої інформаційної системи «Електронна медична картка пацієнта». В дослідженні наведено загальну характеристику електронної медичної картки, обрано архітектуру та технологію розробки. З функціональної точки зору наведено список функцій використовуваних програмних засобів. Розроблено керівництво користувача по взаємодії з інтерфейсом програмного засобу.

В результаті проведеного дослідження розроблена веб-орієнтована інформаційна система «Електронна медична картка пацієнта». Розроблена система реалізує функції реєстрації картки пацієнта, також можливості внесення змін до картки пацієнта, у випадку необхідності видалення цієї картки.

Інформаційна система розроблена на мові програмування PHP. За базу даних використовувався MySQL.

Проведено розрахунок ефективності від провадження системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Гриценко К. Г. Програма виробничої практики [Текст] / Уклад.: канд. техн. наук, доцент, К.Г.Гриценко, канд. тех. наук, доцент, С.М. Новак. – Суми : УАБС, 2010.

2. Великий тлумачний словник сучасної української мови (з дод та допов.) / Уклад. і голов. ред. В. Т. Бусел. – К.: Ірпінь: ВТФ “Перун”, 2007. – 1736 с.

3. Про захист персональних даних [Електронний ресурс] / Режим доступу: http://zakon4.rada.gov.ua/laws/show/2297-17. – Заголовок з екрану.

4. LINQ to XML [Електронний ресурс] / Режим доступу: http://msdn.microsoft.com/en-us/library/bb387098.aspx. – Заголовок з екрану.

5. Нові правила отримання житлової субсидії [Електронний ресурс] / Режим доступу: https://glavcom.ua/publications/subsidiji-za-novimi-pravilami-yak-rozrahuvati-ta-otrimati-dopomogu-vid-derzhavi-351680.html. – Заголовок з екрану.

6. Електронна послуга: Призначення житлової субсидії [Електронний ресурс] / Режим доступу: https://subsidii.mlsp.gov.ua/. – Заголовок з екрану.

7. Все о субсидиях в Украине на 2017-2018 [Електронний ресурс] / Режим доступу: https://24tv.ua/ru/subsidija_2017_2018_ukraina_online_dokumenty_kalkuljator_kak_ofomit_subsidii_n882033/. – Заголовок з екрану.

8. Юредичний портал України [Електронний ресурс] / Режим доступу:http://www.lawportal.com.ua/zhitlova-subsidija-novi-pravila.html.– Заголовок з екрану.

9. Заповнення декларацію доходів [Електронний ресурс] / Режим доступу:https://subsidii.mlsp.gov.ua/help/dec/zapovnennya_deklaratsii_pro_dokhodi_i_vitrati_osib__yaki_zvernulisya_za_priznachennyam_zhitlovoi_subsidii.htm. – Заголовок з екрану.

10. Законодавство України [Електронний ресурс] / Режим доступу: http://zakon3.rada.gov.ua/laws/show/848-95-п. – Заголовок з екрану.

11. Податкова соціальна пільга [Електронний ресурс] / Режим доступу: https://byhgalter.com/kazakova-podatkova-socialna-pilga-2018-shho-novogo/. – Заголовок з екрану.

12. Опис стандарту IDEF0 | EasyCode [Електронний ресурс] / Режим доступу: http://easy-code.com.ua/2011/03/opis-standartu-idef0. – Заголовок з екрану.

13. Нотація IDEF0 [Електронний ресурс] / Режим доступу: http://www.businessstudio.ru/wiki/docs/v4/doku.php/ru/csdesign/bpmodeling/idef0. – Заголовок з екрану.

14. Laravel Installation [Електронний ресурс] / Режим доступу: https://laravel.com/docs/5.5/installation#installation. – Заголовок з екрану.

15. Laravel [Електронний ресурс] / Режим доступу: https://laracasts.com/skills/laravel. – Заголовок з екрану.

16. How To Install MySQL on Ubuntu 16.04 [Електронний ресурс] / Режим доступу: https://www.digitalocean.com/community/tutorials/how-to-install-mysql-on-ubuntu-16-04. – Заголовок з екрану.

17. MySql [Електронний ресурс] / Режим доступу: http://znaimo.com.ua/MySQL. – Заголовок з екрану.

18. Bringing MySQL to the web [Електронний ресурс] / Режим доступу: https://www.phpmyadmin.net/. – Заголовок з екрану.

19. The Apache Software Foundation [Електронний ресурс] / Режим доступу: https://blogs.apache.org/foundation/entry/apache-is-open. – Заголовок з екрану.

20. How To Move an Apache Web Root to a New Location on Ubuntu 16.04 [Електронний ресурс] / Режим доступу: https://www.digitalocean.com/community/tutorials/how-to-move-an-apache-web root-to-a-new-location-on-ubuntu-16-04. – Заголовок з екрану.

21. How To Install the Apache Web Server on Ubuntu 16.04 [Електронний ресурс] / Режим доступу: https://www.digitalocean.com/community/tutorials/how-to-install-the-apache-web-server-on-ubuntu-16-04. – Заголовок з екрану.

22. Mapping URLs to Filesystem Locations [Електронний ресурс] / Режим доступу: https://docs.phpmyadmin.net/en/latest/index.html. – Заголовок з екрану.

23. PHP - Taking the world by storm [Електронний ресурс] / Режим доступу: http://www.phpbbhq.com/developmentofphp.php. – Заголовок з екрану.

24. PhpStorm Early Access Program [Електронний ресурс] / Режим доступу: https://www.jetbrains.com/phpstorm/eap/. – Заголовок з екрану.

25. Docker Support in PhpStorm [Електронний ресурс] / Режим доступу: https://confluence.jetbrains.com/display/PhpStorm/Docker+Support+in+PhpStorm. – Заголовок з екрану.

26. Documentation PHP [Електронний ресурс] / Режим доступу:

27. How to Install and Configure PHP 7.0 [Електронний ресурс] / Режим доступу: https://www.vultr.com/docs/how-to-install-and-configure-php-70-or-php-71-on-ubuntu-16-04. – Заголовок з екрану.

28. JavaScript - Document Object Model or DOM [Електронний ресурс]/ Режим доступу:https://www.tutorialspoint.com/javascript/javascript_html_dom.htm. – Заголовок з екрану.

29. Documentation Jquery [Електронний ресурс] / Режим доступу: https://learn.jquery.com/. – Заголовок з екрану.

30. The Progressive JavaScript Framework [Електронний ресурс] / Режим доступу: https://vuejs.org/. – Заголовок з екрану.

31. Установить Linux, Apache, MySQL, PHP [Електронний ресурс] / Режим доступу: https://www.digitalocean.com/community/tutorials/linux-apache-mysql-php-lamp-ubuntu-16-04-ru– Заголовок з екрану.

32. LAMP [Електронний ресурс] / Режим доступу: https://uk.wikipedia.org/wiki/LAMP– Заголовок з екрану.

33. Official Ubuntu Documentation [Електронний ресурс] / Режим доступу: https://help.ubuntu.com/– Заголовок з екрану.

34. RussianDocumentation Ubuntu [Електронний ресурс] / Режим доступу: https://help.ubuntu.com/community/RussianDocumentation/– Заголовок з екрану.

35. Настройка веб сервера Apache через Htaccess [Електронний ресурс] / Режим доступу: http://htaccess.net.ru/– Заголовок з екрану.

36. Основы .htaccess на примерах [Електронний ресурс] / Режим доступу: https://habrahabr.ru/post/31054/– Заголовок з екрану.

37. Bootstrap[Електроннийресурс]/Режимдоступу: https://getbootstrap.com/– Заголовок з екрану.

38. HTML [Електронний ресурс] / Режим доступу: http://devdocs.io/html/– Заголовок з екрану.

39. Learn HTML [Електронний ресурс] / Режим доступу: https://www.codecademy.com/learn/learn-html– Заголовок з екрану.

40. CSS Reference [Електронний ресурс] / Режим доступу: https://www.w3schools.com/cssref/default.asp– Заголовок з екрану.

Додаток А

SUMMARY

Berkut I. V. Web-based Information System «Electronic Medical Card of the Patient» – Masters-level Qualification Thesis. Sumy State University, Sumy, 2017.

The essence of the registration of an electronic medical card of the patient was investigated in the work. The analysis of the main factors that influence the solution of this problem is carried out. The main purpose of this study is to develop an automated system for registering an electronic medical card of the patient.

Key words: electronic medical card, PHP, website, development.

АНОТАЦІЯ

Беркут І. В. Веб-орієнтована інформаційна система «Електронна медична картка пацієнта». – Кваліфікаційна магістерська робота. Сумський державний університет, Суми, 2017 р.

У роботі досліджено сутність реєстрації електронної медичної картки пацієнта. Проведений аналіз основних факторів, які впливають на розв’язання даної задачі. Основною метою цього дослідження є розробка автоматизованої системи реєстрації електронної медичної картки пацієнта.

Ключові слова: електронна медична картка, PHP, веб-сайт, розробка.

Додаток Б

Файл Kernel.php

```php
<?php

namespace App\Console;

use Illuminate\Console\Scheduling\Schedule;
use Illuminate\Foundation\Console\Kernel as ConsoleKernel;

class Kernel extends ConsoleKernel
{
    /**
     * The Artisan commands provided by your application.
*
     * @var array
     */
    protected $commands = [
        //
    ];

    /**
     * Define the application's command schedule.
*
     * @param  \Illuminate\Console\Scheduling\Schedule  $schedule
     * @return void
     */
    protected function schedule(Schedule $schedule)
    {
        // $schedule->command('inspire')
        //          ->hourly();
    }

    /**
     * Register the commands for the application.
*
     * @return void
     */
    protected function commands()
    {
        $this->load(__DIR__.'/Commands');

        require base_path('routes/console.php');
    }
}
```

Додаток В

Файл Handler.php

```php
<?php

namespace App\Exceptions;

use Exception;
use Illuminate\Foundation\Exceptions\Handler as ExceptionHandler;

class Handler extends ExceptionHandler
{
    /**
     * A list of the exception types that are not reported.
*
     * @var array
     */
    protected $dontReport = [
        //
    ];

    /**
     * A list of the inputs that are never flashed for
validation exceptions.
*
     * @var array
     */
    protected $dontFlash = [
        'password',
        'password_confirmation',
    ];

    /**
     * Report or log an exception.
*
     * This is a great spot to send exceptions to Sentry, Bugsnag, etc.
*
     * @param  \Exception  $exception
     * @return void
     */
    public function report(Exception $exception)
    {
        parent::report($exception);
    }

    /**
     * Render an exception into an HTTP response.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  \Exception  $exception
```

```
     * @return
     \Illuminate\Http\Response */
    public function render($request, Exception $exception)
    {
        return parent::render($request, $exception);
    }
}
```

Додаток Г

Файл ForgotPasswordController.php

```php
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use Illuminate\Foundation\Auth\SendsPasswordResetEmails;

class ForgotPasswordController extends
Controller {
    /*
    |--------------------------------------------------------------------------
    | Password Reset Controller
    |--------------------------------------------------------------------------
    |
    | This controller is responsible for handling password reset emails
and
    | includes a trait which assists in sending these notifications from
    | your application to your users. Feel free to explore this trait. |

    */

    use SendsPasswordResetEmails;

    /**
     * Create a new controller instance.
*
     * @return void
     */
    public function __construct()
    {
        $this->middleware('guest');
    }
}
```

Додаток Ґ

Файл LoginController.php

```
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use Illuminate\Foundation\Auth\AuthenticatesUsers;

class LoginController extends
Controller {
    /*
    |--------------------------------------------------------------------------
    | Login Controller
    |--------------------------------------------------------------------------
    |
    | This controller handles authenticating users for the application
and
    | redirecting them to your home screen. The controller uses a trait
    | to conveniently provide its functionality to your applications. |

    */

    use AuthenticatesUsers;

    /**
     * Where to redirect users after login.
*
     * @var string
     */
  //  protected $redirectTo = '/home';

    public function redirectTo(){
        return '/profile/'.auth()->id();
    }

    /**
     * Create a new controller instance.
*
     * @return void
     */
    public function __construct()
    {
        $this->middleware('guest')->except('logout');
    }
}
```

Додаток Д

Файл RegisterController.php

```
<?php

namespace App\Http\Controllers\Auth;

use App\User;
use App\Http\Controllers\Controller;
use Illuminate\Support\Facades\Validator;
use Illuminate\Foundation\Auth\RegistersUsers;

class RegisterController extends Controller
{
    /*
    |--------------------------------------------------------------------------
    | Register Controller
    |--------------------------------------------------------------------------
    |
    | This controller handles the registration of new users as well as their
    | validation and creation. By default this controller uses a trait to
    | provide this functionality without requiring any additional code.
    |
    */

    use RegistersUsers;

    /**
     * Where to redirect users after registration.
     *
     * @var string
     */
    //protected $redirectTo = '/home';
    public function redirectTo(){
        return '/profile/'.auth()->id();
    }
    /**
     * Create a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('guest');
    }
```

```
    /**
     * Get a validator for an incoming registration request.
*
     * @param  array  $data
     * @return
     \Illuminate\Contracts\Validation\Validator */
    protected function validator(array $data)
    {
        return Validator::make($data, [
            'name' => 'required|string|max:255',
            'email' => 'required|string|email|max:255|unique:users',
            'password' => 'required|string|min:6|confirmed',
        ]);
    }

    /**
     * Create a new user instance after a valid registration.
*
     * @param  array  $data
     * @return \App\User
     */
    protected function create(array $data)
    {
        $user =  User::create([
            'name' => $data['name'],
            'email' => $data['email'],
            'password' => bcrypt($data['password']),

        ]);

        $user->profile()->create();
        return $user;
    }

    protected function handleUserWasAuthenticated(Request
$request, $throttles)
    {
        if ($throttles) {
            $this->clearLoginAttempts($request);
        }

        if (method_exists($this, 'authenticated')) {
            return $this->authenticated($request, Auth::guard($this-
>getGuard())->user());
        }

        return redirect()->intended($this->redirectPath());
    }

}
```

Додаток Е

Файл ResetPasswordController.php

```php
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use Illuminate\Foundation\Auth\ResetsPasswords;

class ResetPasswordController extends Controller
{
    /*
    |--------------------------------------------------------------------------
    | Password Reset Controller
    |--------------------------------------------------------------------------
    |
    | This controller is responsible for handling password reset requests
    | and uses a simple trait to include this behavior. You're free to
    | explore this trait and override any methods you wish to tweak. |
    |
    */

    use ResetsPasswords;

    /**
     * Where to redirect users after resetting their password.
     *
     * @var string
     */
    protected $redirectTo = '/home';

    /**
     * Create a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('guest');
    }
}
```

Додаток Є

Файл CostController.php

```php
<?php

namespace App\Http\Controllers;

use App\Http\Requests\CostsRequest;
use Illuminate\Http\Request;
use App\Costs;
use Illuminate\Support\Facades\Auth;

class CostsController extends
Controller {
    /**
     * Display a listing of the resource.
*
     * @return
     \Illuminate\Http\Response */
    public function index()
    {
        $costs = Costs::where('user_id','=', Auth::id())->get(); return
        view('/interface.page.costs.index', compact('costs'));
    }

    /**
     * Show the form for creating a new resource.
*
     * @return \Illuminate\Http\Response
     */
    public function create()
    {
        return view('/interface.page.costs.create');
    }

    /**
     * Store a newly created resource in storage.
*
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function store(CostsRequest $request)
    {
        $revenues = new Costs($request->all());
        $revenues->name = $request->name;
        $revenues->kind = $request->kind;
        $revenues->cost_property = $request-
        >cost_property; $revenues->data_costs = $request-
        >data_costs; $revenues->user_id = Auth::id();
        $revenues->save();
```

```
        return redirect('/sencs');
    }


    /**
     * Show the form for editing the specified resource.
*
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function edit($id)
    {

        $costs = Costs::find($id);
        if(!$costs){
            abort(404);
        }
        return view('/interface.page.costs.edit', compact('costs'));
    }

    /**
     * Update the specified resource in storage.
*
     * @param  \Illuminate\Http\Request  $request
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function update(CostsRequest $request, $id)
    {
        Costs::find($id)->update($request-
        >all()); return redirect('/costs');
    }

    /**
     * Remove the specified resource from storage.
*
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function destroy($id)
    {
        $costs = Costs::find($id);
        if(!$costs){
            abort(404);
        }
        $costs->delete();
        return redirect('/costs');
    }
}
```

Додаток Ж

Файл DataUserController.php

```
<?php
namespace App\Http\Controllers;
use App\Http\Requests\DataUsersRequest;
use App\DataUser;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class DataUsersController extends
Controller {
    /**
     * Display a listing of the resource.
*
     * @return
     \Illuminate\Http\Response */
    public function index()
    {
        $datas = DataUser::where('user_id','=', Auth::id())-
        >get(); return view('/interface.page.data_user.index',
compact('datas'));
    }

    /**
     * Show the form for creating a new resource.
*
     * @return \Illuminate\Http\Response
     */
    public function create()
    {
        return view('/interface.page.data_user.create');
    }

    /**
     * Store a newly created resource in storage.
*
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function store(DataUsersRequest $request)
    {
        $data = new DataUser($request->all());
        $data->structure_unit = $request->structure_unit;
        $data->name = $request->name;
        $data->place_of_residence = $request->place_of_residence;
        $data->phone = $request->phone;
        $data->pasport_seria = $request->pasport_seria;
        $data->issuance_pasport = $request->issuance_pasport;
        $data->card_taxes = $request->card_taxes;
```

```
        $data->user_id = Auth::id();
        $data->save();
        return redirect('/housing_communals/create');
    }

    /**
     * Show the form for editing the specified resource.
*
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function edit($id)
    {
        $data = DataUser::find($id);
        if(!$data){
            abort(404);
        }
        return view('/interface.page.data_user.edit', compact('data'));
    }

    /**
     * Update the specified resource in storage.
*
     * @param  \Illuminate\Http\Request  $request
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function update(DataUsersRequest $request, $id)
    {
        DataUser::find($id)->update($request-
        >all()); return redirect('/data_users');
    }

    /**
     * Remove the specified resource from storage.
*
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function destroy($id)
    {
        $data = DataUser::find($id);
        if(!$data){
            abort(404);
        }
        $data->delete();
        return redirect('/data_users');
    }
}
```

Додаток 3

Файл HomeController.php

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class HomeController extends
Controller {
    /**
     * Create a new controller instance.
*
     * @return void
     */
    public function __construct()
    {
        $this->middleware('auth');
    }

    /**
     * Show the application dashboard.
*
     * @return
     \Illuminate\Http\Response */
    public function index()
    {
        return view('home');
    }
}
```

Додаток И

Файл HousingCommunalController.php

```php
<?php

namespace App\Http\Controllers;

use App\HousingCommunal;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use App\Http\Requests\HousingCommunalRequest;

class HousingCommunalsController extends Controller
{
    /**
     * Display a listing of the resource.
*
     * @return
     \Illuminate\Http\Response */
    public function index()
    {
        $comunals = HousingCommunal::where('user_id','=', Auth::id())-
>get();
        return view('/interface.page.housing_communal.index',
compact('comunals'));
    }

    /**
     * Show the form for creating a new resource.
*
     * @return \Illuminate\Http\Response
     */
    public function create()
    {
        return view('/interface.page.housing_communal.create');
    }

    /**
     * Store a newly created resource in storage.
*
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function store(HousingCommunalRequest $request)
    {
        $comunal = new HousingCommunal($request->all());
        $comunal->house = $request->house;
```

```
        $comunal->gas = $request->gas;
        $comunal->cold_water = $request->cold_water;
        $comunal->hot_water = $request->hot_water;
        $comunal->drainage = $request->drainage;
        $comunal->centralized_heating = $request->centralized_heating;
        $comunal->electricity_supply = $request->electricity_supply;
        $comunal->household_waste_removal = $request-
>household_waste_removal;
        $comunal->user_id = Auth::id();
        $comunal->save();
        return redirect('/revenues/create');
    }


    /**
     * Show the form for editing the specified resource.
*
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function edit($id)
    {
        $comunal = HousingCommunal::find($id);
        if(!$comunal){
            abort(404);
        }
        return view('/interface.page.housing_communal.edit',
compact('comunal'));
    }

    /**
     * Update the specified resource in storage.
*
     * @param  \Illuminate\Http\Request  $request
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function update(HousingCommunalRequest $request,
    $id) {
        HousingCommunal::find($id)->update($request-
        >all()); return redirect('/housing_communals');
    }

    /**
     * Remove the specified resource from storage.
*
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function destroy($id)
```

```
    {
        $comunal = HousingCommunal::find($id);
        if(!$comunal){
            abort(404);
        }
        $comunal->delete();
        return redirect('/housing_communals');
    }
}
```

Додаток І

Файл ProfileController.php

```php
<?php

namespace App\Http\Controllers;

use App\Profile;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class ProfilesController extends Controller
{
    public  function show($id){
        $profile = Profile::with('user')->where('user_id', $id)->get();
        if(!empty($profile)){
            return view('/interface.page.profile', compact('profile'));
        }
        else{
            abort(404);
        }

    }

    public function sencs(){
        return view('/interface.page.secncs');
    }
}
```

Додаток Ї

Файл RevenuesController.php

```php
<?php

namespace App\Http\Controllers;

use App\Http\Requests\RevenuesRequvest;
use Illuminate\Http\Request; use
App\Revenues;
use Illuminate\Support\Facades\Auth;

class RevenuesController extends Controller
{
    /**
     * Display a listing of the resource.
*
     * @return
     \Illuminate\Http\Response */
    public function index()
    {
        $revenues = Revenues::where('user_id','=', Auth::id())-
        >get(); return view('/interface.page.revenues.index',
compact('revenues'));
    }

    /**
     * Show the form for creating a new resource.
*
     * @return \Illuminate\Http\Response
     */
    public function create()
    {
        return view('/interface.page.revenues.create');
    }

    /**
     * Store a newly created resource in storage.
*
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function store(RevenuesRequvest $request)
    {
        $revenues = new Revenues($request->all());
        $revenues->full_name = $request->full_name;
        $revenues->birth = $request->birth;
```

```
        $revenues->card = $request->card;
        $revenues->type_of_income = $request->type_of_income;
        $revenues->amount_income = $request->amount_income;
        $revenues->source_income = $request->source_income;
        $revenues->user_id = Auth::id(); $revenues->save();

        return redirect('/costs/create');
    }


    /**
     * Show the form for editing the specified resource.
*
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function edit($id)
    {
        $revenues = Revenues::find($id);
        if(!$revenues){
            abort(404);
        }
        return view('/interface.page.revenues.edit',
compact('revenues'));
    }

    /**
     * Update the specified resource in storage.
*
     * @param  \Illuminate\Http\Request  $request
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function update(Request $request, $id)
    {
        Revenues::find($id)->update($request-
        >all()); return redirect('/revenues');
    }

    /**
     * Remove the specified resource from storage.
*
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function destroy($id)
    {
        $revenues = Revenues::find($id);
        if(!$revenues){
            abort(404);
```

```
            }
            $revenues->delete();
            return redirect('/revenues');
        }
}
```

Додаток Й

Файл Profile.php

```php
<?php

namespace App\Http\Middleware;

use Closure;


class Profile
{
    /**
     * Handle an incoming request.
*
     * @param  \Illuminate\Http\Request  $request
     * @param  \Closure  $next
     * @return mixed
     */
    public function handle($request, Closure $next)
    {
        $article = \App\Profile::find($request->parameter('id'));

        if ($article->user_id != Auth::user()->id)
        {
            abort(404);
        }
        return $next($request);
    }
}
```

Додаток К

Файл CostsRequest.php

```php
<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class CostsRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
*
     * @return bool
     */
    public function authorize()
    {
        return true;
    }

    /**
     * Get the validation rules that apply to the request.
*
     * @return array
     */
    public function rules()
    {
        return [
            'name' => 'required',
            'kind' => 'required',
            'cost_property' => 'required',
            'data_costs' => 'required',
        ];
    }
}
```

Додаток Л

Файл DataUsersRequest.php

```php
<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class DataUsersRequest extends
FormRequest {
    /**
     * Determine if the user is authorized to make this request.
*
     * @return bool
     */
    public function authorize()
    {
        return true;
    }

    /**
     * Get the validation rules that apply to the request.
*
     * @return array
     */
    public function rules()
    {
        return [
            'structure_unit' => 'required',
            'name' => 'required',
            'place_of_residence' => 'required',
            'phone' => 'required',
            'pasport_seria' => 'required',
            'issuance_pasport' => 'required',
            'card_taxes' => 'required',
        ];
    }
}
```

Додаток М

Файл HousingCommunalRequest.php

```php
<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class HousingCommunalRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
*
     * @return bool
     */
    public function authorize()
    {
        return true;
    }

    /**
     * Get the validation rules that apply to the request.
*
     * @return array
     */
    public function rules()
    {
        return [
            'house' => 'required',
            'gas' => 'required',
            'cold_water' => 'required',
            'hot_water' => 'required',
            'drainage' => 'required',
            'centralized_heating' => 'required',
            'electricity_supply' => 'required',
            'household_waste_removal' => 'required',
        ];
    }
}
```

Додаток Н

Файл RevenuesRequvest.php

```php
<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class RevenuesRequvest extends
FormRequest {
    /**
     * Determine if the user is authorized to make this request.
*
     * @return bool
     */
    public function authorize()
    {
        return true;
    }

    /**
     * Get the validation rules that apply to the request.
*
     * @return array
     */
    public function rules()
    {
        return [
            'full_name' => 'required',
            'birth' => 'required',
            'card' => 'required',
            'type_of_income' => 'required',
            'amount_income' => 'required',
            'source_income' => 'required',
        ];
    }
}
```

Додаток О

Файл AuthServiceProvider.php

```php
<?php

namespace App\Providers;

use Illuminate\Support\Facades\Gate;
use Illuminate\Foundation\Support\Providers\AuthServiceProvider as
ServiceProvider;

class AuthServiceProvider extends
ServiceProvider {
    /**
     * The policy mappings for the application.
*
     * @var array
     */
    protected $policies = [
        'App\Model' => 'App\Policies\ModelPolicy',
    ];

    /**
     * Register any authentication / authorization services.
*
     * @return void
     */
    public function boot()
    {
        $this->registerPolicies();

        //
    }
}
```

Додаток П

Файл RouteServiceProvider.php

```php
<?php

namespace App\Providers;

use Illuminate\Support\Facades\Route;
use Illuminate\Foundation\Support\Providers\RouteServiceProvider
as ServiceProvider;

class RouteServiceProvider extends
ServiceProvider {
    /**
     * This namespace is applied to your controller routes.
*
     * In addition, it is set as the URL generator's root namespace.
*
     * @var string
     */
    protected $namespace = 'App\Http\Controllers';

    /**
     * Define your route model bindings, pattern filters, etc.
*
     * @return void
     */
    public function boot()
    {
        //

        parent::boot();
    }

    /**
     * Define the routes for the application.
*
     * @return void
     */
    public function map()
    {
        $this->mapApiRoutes();

        $this->mapWebRoutes();

        //
    }
```

```
    /**
     * Define the "web" routes for the application.
*
     * These routes all receive session state, CSRF protection, etc.
*
     * @return void
     */
    protected function mapWebRoutes()
    {
        Route::middleware('web')
             ->namespace($this->namespace)
             ->group(base_path('routes/web.php'));
    }

    /**
     * Define the "api" routes for the application.
*
     * These routes are typically stateless.
     *
     * @return
     void */
    protected function mapApiRoutes()
    {
        Route::prefix('api')
             ->middleware('api')
             ->namespace($this->namespace)
             ->group(base_path('routes/api.php'));
    }
}
```

Додаток Р

Файл Costs.php

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Costs extends Model
{
    protected $fillable = [
        'name',
        'kind',
        'cost_property',
        'data_costs',
        'user_id',
    ];
}
```

Додаток С

Файл DataUser.php

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class DataUser extends Model
{
    protected $fillable = [
        'structure_unit',
        'name',
        'place_of_residence',
        'phone',
        'pasport_seria',
        'issuance_pasport',
        'card_taxes',
        'user_id',
    ];
}
```

Додаток Т

Файл HousingCommunal.php

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class HousingCommunal extends Model
{
    protected $fillable = [
        'house',
        'gas',
        'cold_water',
        'hot_water',
        'drainage',
        'centralized_heating',
        'electricity_supply',
        'household_waste_removal',
        'user_id',
    ];
}
```

Додаток У

Файл Profile.php

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Profile extends Model
{
    protected $fillable = [
        'user_id',
    ];

    public function user()
    {
        return $this->belongsTo(User::class, 'user_id');
    }
}
```

Додаток Ф

Файл Revenues.php

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Revenues extends Model
{
    protected $table = "revenues";
    protected $fillable = [
        'full_name',
        'birth',
        'card',
        'type_of_income',
        'amount_income',
        'source_income',
        'user_id',
    ];
}
```

Додаток Х

Файл User.php

```php
<?php

namespace App;

use Illuminate\Notifications\Notifiable;
use Illuminate\Foundation\Auth\User as Authenticatable;

class User extends Authenticatable
{
    use Notifiable;

    /**
     * The attributes that are mass assignable.
*
     * @var array
     */
    protected $fillable = [
        'name', 'email', 'password',
    ];

    /**
     * The attributes that should be hidden for arrays.
*
     * @var array
     */
    protected $hidden = [
        'password', 'remember_token',
    ];


    public function profile()
    {
        return $this->hasOne(Profile::class, 'user_id', 'id');
    }
}
```

Додаток Ц

Файл CreateUsersTable.php

```php
<?php

use Illuminate\Support\Facades\Schema; use
Illuminate\Database\Schema\Blueprint; use
Illuminate\Database\Migrations\Migration;

class CreateUsersTable extends
Migration {
    /**
     * Run the migrations.
*
     * @return void
     */
    public function up()
    {
        Schema::create('users', function (Blueprint $table) {
            $table->increments('id');
            $table->string('name');
            $table->string('email')->unique();
            $table->string('password');
            $table->rememberToken();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
*
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('users');
    }
}
```

Додаток Ч

Файл CreatePasswordResetsTable.php

```php
<?php

use Illuminate\Support\Facades\Schema; use
Illuminate\Database\Schema\Blueprint; use
Illuminate\Database\Migrations\Migration;

class CreatePasswordResetsTable extends
Migration {
    /**
     * Run the migrations.
*
     * @return void
     */
    public function up()
    {
        Schema::create('password_resets', function (Blueprint $table) {
            $table->string('email')->index();
            $table->string('token');
            $table->timestamp('created_at')->nullable();
        });
    }

    /**
     * Reverse the migrations.
*
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('password_resets');
    }
}
```

Додаток Ш

Файл CreateProfilesTable.php

```php
<?php

use Illuminate\Support\Facades\Schema; use
Illuminate\Database\Schema\Blueprint; use
Illuminate\Database\Migrations\Migration;

class CreateProfilesTable extends Migration
{
    /**
     * Run the migrations.
*
     * @return void
     */
    public function up()
    {
        Schema::create('profiles', function (Blueprint $table) {
            $table->increments('id');
            $table->integer('user_id');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
*
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('profiles');
    }
}
```

Додаток Щ

Файл CreateDataUsersTable.php

```php
<?php

use Illuminate\Support\Facades\Schema; use
Illuminate\Database\Schema\Blueprint; use
Illuminate\Database\Migrations\Migration;

class CreateDataUsersTable extends
Migration {
    /**
     * Run the migrations.
*
     * @return void
     */
    public function up()
    {
        Schema::create('data_users', function (Blueprint $table) {
            $table->increments('id');
            $table->string('structure_unit');
            $table->string('name');
            $table->string('place_of_residence');
            $table->string('phone');
            $table->string('pasport_seria');
            $table->string('issuance_pasport');
            $table->string('card_taxes');
            $table->integer('user_id');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
*
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('data_users');
    }
}
```

Додаток Ю

Файл CreateHousingCommunalsTable.php

```php
<?php

use Illuminate\Support\Facades\Schema; use
Illuminate\Database\Schema\Blueprint; use
Illuminate\Database\Migrations\Migration;

class CreateHousingCommunalsTable extends Migration
{
    /**
     * Run the migrations.
*
     * @return void
     */
    public function up()
    {
        Schema::create('housing_communals', function (Blueprint $table)
{
            $table->increments('id');
            $table->string('house');
            $table->string('gas');
            $table->string('cold_water');
            $table->string('hot_water');
            $table->string('drainage');
            $table->string('centralized_heating');
            $table->string('electricity_supply');
            $table->string('household_waste_removal');
            $table->integer('user_id');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
*
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('housing_communals');
    }
}
```

Додаток Я

Файл CreateRevenuesTable.php

```php
<?php

use Illuminate\Support\Facades\Schema; use
Illuminate\Database\Schema\Blueprint; use
Illuminate\Database\Migrations\Migration;

class CreateRevenuesTable extends Migration
{
    /**
     * Run the migrations.
*
     * @return void
     */
    public function up()
    {
        Schema::create('revenues', function (Blueprint $table) {
            $table->increments('id');
            $table->string('full_name');
            $table->string('birth');
            $table->string('card');
            $table->string('type_of_income');
            $table->string('amount_income');
            $table->string('source_income');
            $table->integer('user_id');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
*
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('revenues');
    }
}
```

Додаток АБ

Файл CreateCostsTable.php

```php
<?php

use Illuminate\Support\Facades\Schema; use
Illuminate\Database\Schema\Blueprint; use
Illuminate\Database\Migrations\Migration;

class CreateCostsTable extends
Migration {
    /**
     * Run the migrations.
*
     * @return void
     */
    public function up()
    {
        Schema::create('costs', function (Blueprint $table) {
            $table->increments('id');
            $table->string('name');
            $table->string('kind');
            $table->string('cost_property');
            $table->string('data_costs');
            $table->integer('user_id');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
*
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('costs');
    }
}
```

Додаток АВ

Файл style.css

```
header.main-header .header-container
    { margin: 0 auto;
    padding: 60px 0px;
    background-color: rgb(193, 207, 77);
    font-family: "PTSansCaptionBold";
    background-image: url(../image/logo.png);
    background-repeat: no-repeat;
    background-position: 35px 37px;
    background-size: 140px;
}

header.main-header .text {
    padding-left: 190px;
    text-shadow: 0 -2px 3px rgba(0, 3, 1, 0.64);
    color: #fff;
}

.show-title{
    text-align: center;
}

.name-profile{
    text-transform: uppercase;
}

.title-end{
    text-align: center;
}

.mini-tite{
    text-align: center;
}

.title-index{
```

```
    text-align: center;
    margin-bottom: 40px;
}

.link-style a{
    font-size: 30px;
}

.link-style{
    margin-top: 40px;
}

.new_footer {
    margin-top: 40px;
    width: 100%;
    height: 80px;
    background: #c1cf4d;
}

.footer_logo {
    text-align: center;
    padding-top: 1%;
}

.footer_logo_img {
    margin: auto;
}

.navbar-default {
    background-color: #c1cf4d;
    border-color: #e7e7e7;
}

.navbar-default .navbar-brand{
    color: #ffffff;
}

.navbar-default .navbar-brand:hover {
    color: #e7e7e7;
```

```
    background-color: transparent;
}

.navbar-default .navbar-nav>li>a {
    color: #ffffff;
}

.navbar-default .navbar-nav>li>a:hover {
    color: #e7e7e7;
}
.footer_logo_img {
    width: 60px;
}
```

Додаток АГ

Файл index.php

```php
<?php

/**
 *
 * @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, then it
| will load up this application so that we can run it and
send | 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);
```

Додаток АҐ

Файл bootstrap.js

```
window._ = require('lodash');

/**
 * We'll load jQuery and the Bootstrap jQuery plugin which provides support
 * for JavaScript based Bootstrap features such as modals and tabs. This
 * code may be modified to fit the specific needs of your application.
 */

try {
    window.$ = window.jQuery = require('jquery');

    require('bootstrap-sass');
} catch (e) {}

/**
 * We'll load the axios HTTP library which allows us to easily issue requests
 * to our Laravel back-end. This library automatically handles sending the
 * CSRF token as a header based on the value of the "XSRF" token cookie.
 */

window.axios = require('axios');

window.axios.defaults.headers.common['X-Requested-With'] = 'XMLHttpRequest';

/**
 * Next we will register the CSRF Token as a common header with Axios so that
```

```
 * all outgoing HTTP requests automatically have it attached. This is just
 * a simple convenience so we don't have to attach every token manually.
 */

let token = document.head.querySelector('meta[name="csrf-token"]');

if (token) {
    window.axios.defaults.headers.common['X-CSRF-TOKEN'] = token.content;
} else {
    console.error('CSRF token not found: https://laravel.com/docs/csrf#csrf-x-
csrf-token');
}

/**
 * Echo exposes an expressive API for subscribing to channels and listening
 * for events that are broadcast by Laravel. Echo and event broadcasting
 * allows your team to easily build robust real-time web applications.
 */

// import Echo from 'laravel-echo'

// window.Pusher = require('pusher-js');

// window.Echo = new Echo({
//     broadcaster: 'pusher',
//     key: 'your-pusher-key',
//     cluster: 'mt1',
//     encrypted: true
// });
```

Додаток АД

Файл master.blade.php

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-
    scale=1"> <meta name="description" content=""> <meta name="author"
    content="">
    {{--<title>@yield('title')</title>--
}}      <title>DIPLOM</title>
    @include('interface.css')
    @yield('css')
</head>
<body>
@include('interface.nav')

@yield('content')

@include('interface.footer')
@include('interface.scripts')
@yield('scripts')
</body>
</html>
```

Додаток АЕ

Файл web.php

```php
<?php

/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These |
routes are loaded by the RouteServiceProvider within a group which |
contains the "web" middleware group. Now create something great!
|
*/

Route::get('/', 'AppController@home');

Auth::routes();

Route::get('/home', 'HomeController@index')->name('home');
Route::get('/sencs', 'ProfilesController@sencs')->name('sencs');
Route::get('/profile/{id}',     'ProfilesController@show')->middleware('auth');
//Route::resource('data_users',     'DataUsersController')->middleware('auth');
Route::resource('data_users', 'DataUsersController', ['except' => [
    'show'
]])->middleware('auth');

Route::resource('housing_communals', 'HousingCommunalsController', ['except' =>
[
    'show'
]])->middleware('auth');

Route::resource('revenues', 'RevenuesController', ['except' => [
    'show'
]])->middleware('auth');

Route::resource('costs', 'CostsController', ['except' => [
    'show'
]])->middleware('auth');
```

Додаток АЄ

Файл .env

```
APP_NAME=Laravel
APP_ENV=local
APP_KEY=base64:Zk4Fc1CzIu1XPiFAoSQ7g4FZyU+xrrsNK1VrsNYFk4c=
APP_DEBUG=true
APP_LOG_LEVEL=debug
APP_URL=http://localhost

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=diplom
DB_USERNAME=root
DB_PASSWORD=

BROADCAST_DRIVER=log
CACHE_DRIVER=file
SESSION_DRIVER=file
SESSION_LIFETIME=120
QUEUE_DRIVER=sync

REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379

MAIL_DRIVER=smtp
MAIL_HOST=smtp.mailtrap.io
MAIL_PORT=2525
MAIL_USERNAME=null
MAIL_PASSWORD=null
MAIL_ENCRYPTION=null

PUSHER_APP_ID=
PUSHER_APP_KEY=
PUSHER_APP_SECRET=
PUSHER_APP_CLUSTER=mt1
```