Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка архітектури АРМ з використанням SQL технологій

Виконав(ла): студент(ка) 4 курсу, групи СПс-41
спеціальності 121 Програмна інженерія
(шифр і назва спеціальності)

(підпис) Кікцьо Н.Ю. (прізвище та ініціали)

Керівник (підпис) Яворська Є.Б. (прізвище та ініціали)

Нормоконтроль (підпис) Стоянов Ю.М. (прізвище та ініціали)

Завідувач кафедри (підпис) Петрик М.Р. (прізвище та ініціали)

Рецензент (підпис) (прізвище та ініціали)

Тернопіль
2024

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет Комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра Програмної інженерії
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Петрик М.Р.
(підпис) (прізвище та ініціали)
« » 2024 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавр
(назва освітнього ступеня)

за спеціальністю 121 Програмна інженерія
(шифр і назва спеціальності)

студенту Кікцьо Назарію Юрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка архітектури АРМ з використанням SQL технологій

Керівник роботи Яворська Євгенія Богданівна, к.т.н., доц.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «_15__» *квітня* 2024 року № *4/7-343*

2. Термін подання студентом завершеної роботи

3. Вихідні дані до роботи Вимоги замовника, технічні умови, технічне завдання

4. Зміст роботи (перелік питань, які потрібно розробити)

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності,	Гурик О.Я., доцент кафедри МТ		
основи охорони праці	Лазарюк В.В., доцент кафедри МТ		

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання		
2	Аналіз завдання		
3	Виконання розділу 1		
4	Виконання розділу 2		
5	Виконання розділу 3		
6	Оформлення пояснювальної записки		
7	Оформлення графічного та презентаційного матеріалу		
8	Попередній захист		
9	Захист		

Студент _____ Кікцьо Н.Ю.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Яворська Є.Б.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

АВТОМАТИЗОВАНА СИСТЕМА ОБЛІКУ, БАЗА ДАНИХ, МЕДИЧНІ ПОСЛУГИ

В процесі виконання роботи розроблено інформаційну систему за допомогою якої автоматизовано процес роботи реєстратури поліклініки, а саме ведення і облік медичних карт пацієнтів. Це уможливить підвищення ефективності роботи реєстратури, спрощення роботи її працівників.

Кваліфікаційна робота бакалавра складається:

с. 55, рис. 27, табл. 3, джер. 18

Проведено тестування, дослідження а також аналіз системи, яка забезпечує достатю інформацію.

Ключові слова: інформаційні технології, програмне забезпечення, інформаційні системи, UML Class Diagram; UML Actvity Diagram, база даних, Django;ABSTRACT

AUTOMATED ACCOUNTING SYSTEM, DATABASE, MEDICAL SERVICES

In the course of the work, an information system was developed, with the help of which the work process of the polyclinic registry is automated, namely, the management and accounting of patients' medical records. This will make it possible to increase the efficiency of the registry office and simplify the work of its employees.

Testing, research and analysis of the system, which provides sufficient information, has been carried out.

Keywords: information technologies, software, information systems, UML Class Diagram; UML Activity Diagram, database, Django;

ЗМІСТ

ВСТУП

Актуальність обраної теми зумовлена тим, що на сьогоднішній день електронний документообіг дедалі більше інтегрується у різні системи. У тому числі й до медичних закладів.

Медичні карти пацієнтів, внутрішні документи реально та доступно перетворювати на електронний вигляд. Таким чином, реєстратура будь-якої поліклініки за лічені хвилини може знайти відомості про прийнятих пацієнтів, виклики, кабінети.

Існуючі системи нині є універсальними. Більшість їх є занадто дорогими, або задовольняють в повному обсязі вимогам.

Мета бакалаврської роботи дослідження предметної галузі та розробка програмного забезпечення.

Потрібне створення повнофункціональної інформаційної системи, використання якої сприятиме підвищенню ефективності роботи поліклініки, переходу на якісно новий рівень обслуговування та лікування пацієнтів. Також система не повинна бути надто дорогою в розробці та обслуговуванні.

Досягнення мети вимагає вирішення таких задач:

- дослідити аналоги інформаційних систем даної предметної галузі;
- провести аналіз завдання;
- визначити вимоги до системи, що розробляється;
- визначити інструментальні засоби розробки;
- розробити базовий інтерфейс веб-додатку.

Система логічно поділяється на дві частини – для пацієнтів та для лікарів. Лікарі можуть керувати даними, пацієнти – отримувати доступ до електронної версії медичних карток.

Проходження лікування пацієнта в поліклініці передбачає:

- оформлення особистої картки пацієнта;
- зЗбереження особистої картки пацієнта, історій всіх хвороб, поставлених діагнозів, результатів проведення досліджень / аналізів;

- направлення пацієнта до лікаря, на проведення дослідження, здачу аналізів;

- оформлення довідок / лікарняних листів.

Система, що розробляється, повинна відповідати завданням:

- спрощення доступу до персональних даних пацієнта;

- доступ пацієнта до своїх особистих даних;

- швидке доведення результатів проведення аналізів пацієнта до лікаря;

- скорочення штатної чисельності відділу реєстратури, витрат на зарплату та скорочення людських та часових витрат на обробку інформації;

- централізоване зберігання всіх даних про пацієнта;

- зменшення кількості суперечливих даних.

1 РОЗРОБКА ПРОГРАМНОГО ЗАСОБУ

1.1. Опис предметної області

Поліклініка - лікувально-профілактичний заклад, який надає спеціалізовану медичну допомогу хворим та хворим удома.

Клініка відрізняється від диспансерів великим обсягом діяльності та більш спеціалізованою медичною допомогою (кількість спеціалізованих та допоміжних приміщень для надання кваліфікованої медичної допомоги в клініці та багато іншого).

Заклади складаються з багатьох відділень - від терапевтичного до функціональної діагностики, залежно від виду поліклініки.

Поліклініка забезпечує:

- медичне та невідкладне лікування гострої та раптової хвороби, травм та отруєнь;

- медичне лікування амбулаторних пацієнтів та пацієнтів удома із збереженням у необхідних випадках госпіталізації;

- проведення антиепідемічних заходів (вакцинація, ідентифікація інфекційних захворювань, динамічний моніторинг осіб, які були в контакті з інфекційними пацієнтами та одужувальні, тривожні, за необхідності, органи влади та ін.).

Робота поліклініки ґрунтується на диспансерному методі (підбір осіб, які підлягають динамічному моніторингу, спеціалізованому лікуванню, систематичному вивченню умов праці та життя населення). У клініці проводяться профілактичні заходи, які у поєднанні з медичною роботою є основними цілями поліклінік та відіграють вирішальну роль у скороченні захворюваності серед населення.

Поліклініка – це динамічний контроль стану здоров'я населення.

В умовах роботи поліклініки доводиться мати справу з великими потоками інформаціями, управляти та змінювати дані про пацієнтів, лікарів, страхових компаній та багато іншого.

1.1.1. Аналог ІС

Electronic Medical Records Software від компанії MicroMD. [4]

MicroMD – це хмарна система управління практикою та електронна медична документація (EMR), призначена для медичних працівників. Він містить інструменти, які допомагають користувачам виконувати повсякденні операції більш ефективно, щоб вони могли досягти успіху у своїй спеціалізації.

Продукт глобального постачальника медичних послуг Генрі Шейн, MicroMD має комплексні рішення для управління практикою для фахівців з усіх спеціальностей та розмірів клінік.

Розширені функції планування, реєстрації пацієнтів, управління персоналом, інтегрованих рішень EDI, детальної звітності, форматування електронного файлу заявок тощо. Як заявляє виробник, все зроблено так, щоб дозволити користувачам максимально оптимізувати робочі процеси. Система проста в освоєнні і може бути масштабована для підтримки потреб у клініці.

ПЗ підходить для широкого кола медичних напрямків, але може не підходить для деяких спеціальностей та типів організації. На даний момент розробники продовжують розширювати та створювати контент та функціональні можливості EMR для задоволення різних потреб у спеціальностях щороку.

Як заявляють розробники, програма допомагає зменшити непотрібну паперову тяганину, підвищує продуктивність клініки та рівень обслуговування клієнтів.

Основні характеристики MicroMD EMR:

- Електронні медичні картки.

- Підтримка E/M кодингу (актуально для США, стандарт ведення медичних карт та обслуговування страхових компаній).

- Сумісність з HIPAA (Закон про переносимість та підзвітність медичного страхування).

- Систему EMR можна налаштувати під себе.

- Налаштовувана клінічна документація.

- Підтримка багатьох ОС.

- Електронний підпис.

- Перегляд та збереження діагностичних та анатомічних зображень.

- Підтримка пацієнтів, оповіщення та нагадування про записи до лікарів.

- Управління зображеннями та анотаціями.

- Управління робочим навантаженням.

- Вбудована адміністративна панель.

- Налагодження взаємодії з аптеками (доставка ліків, виписка рецептів).

- Відкрита архітектура та взаємодії із системою.

- Підключення пристроїв.

- Інтерфейс із цифровими моніторами серцево-легеневої та життєво важливих функцій.

- Використовувати стандартну та настроювану звітність для аналізу тенденцій.

- Дозволяє настроїти вашу електронну медичну систему запису.

- Підтримка звітності.

- Спеціальні системні продумані шаблони.

- Управління записом до лікарів.

- Зручний перегляд клінічних даних пацієнта.

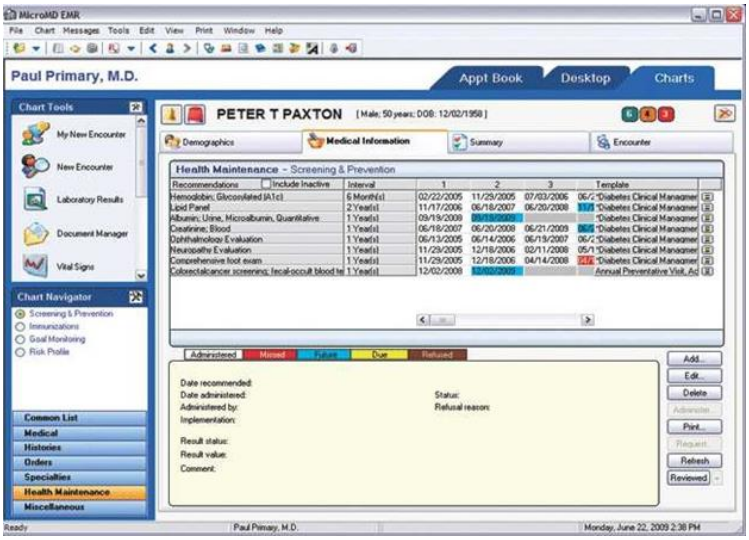

Рисунок 1.1 – Інтерфейс EMR[4]

Плюси підсистеми:

- Великі функціональні можливості;
- Хороший інтерфейс;
- Підтримується законодавчі стандарти (США);
- Перегляд розкладу на сайт.

Мінуси:

- Вартість інтеграції та підтримки;
- Орієнтована на стандарти американської охорони здоров'я;
- Підтримується лише англійська мова;
- Підходить не всім, незрозуміло, як з'ясувати кому саме;
- Тільки декстопна версія.

«Electronic Medical Records» орієнтована на західний ринок медичних послуг, підтримує лише одну мову та дуже дорога.

Відомі системи мають свої переваги та недоліки. Але, жодна з них не має достатньої гнучкості до налаштування, немає інтерфейсу для пацієнтів і немає підтримки веб-версії. Тому я вирішив реалізувати свій аналог системи управління медичними картами пацієнтів, використовуючи сучасні веб-технології.

1.1.2. Аналіз завдання

Сформулюю вимоги до функціоналу застосунку. За результатом зроблено такі висновки щодо вимог веб-програми:

- використовує сучасні мови та фреймворки для тривалої підтримки;
- бути легковбудовуваною і модульною;
- мати бекенд частину (для медичних працівників) та фронтенд частину (для пацієнтів);
- підтримувати необхідний рівень захисту від хакерських атак та кешування;
- маловимоглива до програмних та апаратних засобів;

Основною особливістю даної системи буде доступність та малобюджетна технологія. Також систему можуть легко підтримувати сторонні розробники. ІС придатна для впровадження безлічі функціональних налаштувань, може легко масштабуватися.

Ця система – це прототип для демонстрації того, як можна буде використовувати систему на реальних завданнях. [13, 14, 15, 17]

1.2. Інструментальні засоби для розробки

1.2.1. Python.[5]

Python – об'єктно-орієнтована мова. Програмні модулі та структури даних можна використовувати як об'єкти, тобто. мають властивості та методи.

Підпрограми в Python оформляються лише як функції, але ці функції можуть не повертати значень, а можуть повертати кілька значень як структур даних.

Функції, оформлені за певними правилами, поєднуються в модулі (бібліотеки функцій). Модулі (або деякі функції з модулів) при необхідності підключаються до програм користувача. Такий підхід дозволяє заощаджувати пам'ять обчислювальної системи та не займати її непотрібним кодом.

Філософія Python включає такі правила:

- Гарне краще, ніж потворне.

- Явне краще, ніж неявне.

- Просте краще, ніж складне.

- Читання має значення.

- Повинен існувати один – і, бажано, лише один – очевидний спосіб зробити це.

І мова Python цілком відповідає цим постулатам. Простий, ясний з першого погляду, несуперечливий синтаксис, використання традиційного запису алгебри та звичних за іншими мовами конструкцій, прості засоби для роботи зі складними типами даних, строга модульність і компактність коду, багаті можливості об'єктно-орієнтованого програмування - все це є. А якщо додати сюди ще й високу (зрозуміло, для інтерпретованої мови) швидкодію, найбагатшу стандартну бібліотеку та наявність величезної кількості сторонніх бібліотек, що виконують різні завдання, стає зовсім не дивно, що Python в даний час настільки популярний.

За допомогою цієї мови можна писати настільні програми, у тому числі такі, що мають графічний інтерфейс, системні утиліти, інтернет-програми та командні сценарії для інших програмних пакетів. А ще на Python можна розробляти Web-серверні програми-наприклад, Web-сайти.

Ця мова обрана як основна для розробки, оскільки досить просто можна створювати складні, легко підтримувані програми. І одна з найкращих переваг – розвинена спільнота. Це означає, що якісних фрейворків, сторонніх бібліотек, туторіалів та розв'язання складних завдань достатньо.

1.2.2. Django web framework.[6]

Django - це високого рівня веб-інфраструктура Python, яка дозволить швидко створити безпечні та підтримувані веб-сайти. Django опікується багатьма проблемами веб-розробки, тому можна зосередитися на написанні своєї програми. Це безкоштовне та відкрите джерело, має процвітаючу та активну спільноту, відмінну документацію та безліч опцій для безкоштовної та платної підтримки. [6,11]

Використовує шаблон проектування MVC.

Варто додати, що фреймворк має хорошу підтримку з боку спільноти розробників і наявність в Інтернеті великої кількості сайтів, присвячених Django, на яких можна знайти статті з програмування, відповіді на питання, що часто виникають, і готові приклади коду.

Деякі можливості та особливості Django:

- реалізація архітектури «модель-контролер-шаблон»;
- включає легкий і автономний веб-сервер для розробки та тестування;
- реалізація принципу DRY (Don't Repeat Yourself, не повторюйся), в результаті;
- присутня система серіалізації та перевірки форми, яка може перекладати між HTML-формами та значеннями, що підходять для зберігання у базі даних;
- система інтернаціоналізації, включаючи переклад власних компонентів Django різними мовами;
- підтримка класів проміжного програмного забезпечення (“middleware”), які можуть втручатися на різних етапах обробки запитів та виконувати функції користувача;
- структура кешування, яка може використовувати будь-який із кількох методів кешування;
- інтерфейс до вбудованої платформи тестування Python;

- потужний шаблонізатор, заснований на спеціальних тегах, з можливістю успадкування шаблонів;

- уніфіковані засоби для роботи з базами даних будь-яких форматів, що підтримуються: SQLite, MySQL, PostgreSQL, Oracle, Microsoft SQL Server, Firebird та ін;

- авторизація та аутентифікація, підключення зовнішніх модулів аутентифікації;

- динамічний адміністративний інтерфейс;

- багаті кошти на роботи з формами;

- інструменти для реалізації розмежування доступу;

- вбудований адміністративний сайт із можливостями налаштування, який можна використовувати для роботи з даними;

- проста та зрозуміла структура створюваних сайтів: кожен розділ сайту є окремим додатком, який може бути відчуженим та використаним в іншому сайті; всі додатки об'єднуються в проект, власне, і сайт.

Варто відзначити, що такі відомі сайти, як YouTube, DropBox, Reddit, написані на основі даного фреймворку.

Поряд із перевагами, приходять недоліки. Існує багато недоліків Django, згаданих нижче:

- використовує шаблон маршрутизації, вказуючи його URL-адресу.

- Django занадто монолітний.

- все ґрунтується на Django ORM.

- компоненти розгортаються разом.

- розвивається не так швидко, як інші фреймворки.

Незважаючи на його недоліки, саме цей фреймворк обраний для розробки програми. На його основі можна перетворити простий web сайт в потужний, і при цьому додаток, що легко підтримується.

1.2.3. PostgreSQL.[7]

PostgreSQL – реляційна СУБД, яка основана на теорії множин, реалізована у вигляді двовимірних таблиць, де дані зберігаються рядками, і суворо контролює типи стовпців. Незважаючи на зростаючий інтерес до нових тенденцій у галузі баз даних, реляційний стиль є найпопулярнішим і, ймовірно, залишиться таким ще досить довго.

У PostgreSQL вбудована підтримка Unicode, послідовностей, успадкування таблиць, підзапитів, а реалізація SQL точніше слідує стандарту ANSI, ніж будь-яка інша з представлених на ринку. СУБД [17] є швидкою та надійною, здатна зберігати терабайти даних та довела працездатність у таких високонавантажених проектах, як Skype, французька Національна каса з виплати допомоги багатодітним сім'ям (Caisse Nationale d'Allocations Familiales – CNAF) та Федеральне управління цивільної авіації США.

СУБД підтримує більшу частину стандарту SQL і пропонує наступні функції: формування складних запитів, використання зовнішніх ключів, тригерів, змінюваних уявлень, трансакціної цілісності, існування декількох версій.

Також, користувач, за бажанням, може всіляко розширити можливості PostgreSQL, шляхом створення типів даних, функцій, операторів, агрегатних функцій, методів індексації, процедурних мов.

Враховуючи, що PostgreSQL має вільну ліцензію, то можна використовувати безкоштовно, робити зміни, поширювати для використання для виконання власних, освітніх, комерційних проєктів.

1.2.4. Redis.

Redis (REmote DIctionary Service – віддалена служба словників) – просте у роботі сховище ключів та значень із розвиненим набором команд. Перша версія системи вийшла у 2009 році. І у швидкодії у нього практично немає суперників. Читання відбувається швидко, а запис ще швидше - на деяких еталонних тестах продемонстровано до 100 000 операцій SET за секунду. Автор Redis Сальваторе Санфіліппо (Salvatore Sanfilippo) називає свій проект «сервером структур даних», щоб підкреслити закладені в нього можливості роботи зі складними типами даних та інші особливості.

Використовується для кращого функціонування бази даних Postgres.

Крім сервера структур даних, Redis є ще й блокуючою чергою (або стеком), а також системою публікації-підписки. У ній можна налаштовувати політики терміну зберігання, рівні довговічності та параметри реплікації. Все це робить Redis скоріше комплектом інструментальних засобів, до якого входять корисні алгоритми роботи зі структурами даних та процесами, ніж базою даних якогось певного жанру.

1.2.5. Docker.

Docker - це додаток, що дозволяє легко і швидко запускати інші додатки та процеси в контейнерах (контейнеризація), які подібні до віртуальних машин, але є при цьому портованішими, споживають менше ресурсів, а мають невелику залежність від операційної системи машини-хоста.

Першочергово Docker було розроблено для Linux , в якому використовують функції виділення ресурсу ядра Linux , зокрема групи і простори імен ядер, а файлова система підтримує об’єднання. Ядро Linux підтримує простір імен, як правило, для операційного середовища, забезпечує ізолювання уявлення програми, з врахуванням дерев процесів, мережі, ідентифікатора користувача, змонтованих файлових систем. Забезпечення обмеження ресурсу пам’яті і процесора відбувається за допомогою груп ядра. Бібліотека libcontainer дозволяє безпосередньо використати надані ядром Linux засоби віртуалізації.

За допомогою даного інструменту працює база даних і можна легко розгорнути проект на сервері.

1.2.6. Git – система контролю версії. [8]

Git – це вільна та відкрита контролююча версії розподільна система, яка відстежує зміни у комп'ютерних файлах та координації роботи над цими файлами серед кількох людей. Вона була створена для розробки ядра Linux (2005 р.).

Інструмент в основному використовується для керування вихідним кодом у розробці програмного забезпечення, але його можна використовувати для відслідковування змін у файлах. Як, контролююча версії розподілена система, вона націлена на швидкість, цілісність даних та підтримку розподілених нелінійних робочих процесів.

Основна відмінність між Git та будь-яким іншим VCS (включаючи Subversion) - це те, як Гіт приймає свої дані. Концептуальною більшістю інших систем інформація зберігається як перелік файлових змін. Вони (CVS, Subversion, Perforce, Bazaar тощо) вважають збережену інформаці, набором файлів, і зміни, внесені у кожен файл із часом (це зазвичай описується як управління версіями з урахуванням змін).

Git вважає свої дані більш подібними на низку відбитків дуже малої файлової системи. Кожного рази, про роботі з проектом чи при його збереженні, Git, як правило, фіксує вигляд всіх файлів і зберігає посилання на них.

Відмінна перевага інструментів, подібних до Git є те, що кожен, хто має копію з усім необхідним, може відновити проект. Якщо центральний сервер зламається і всі дані загубляться, будь-яка віддалена копія може стати офіційною, тому що це буде достатньо для відновлення працездатності.

1.3. Розробка програми

1.3.1. Проектування моделей у базі даних.

Веб-програми Django отримують доступ і керують даними через об'єкти Python, які називаються моделями. Моделі визначають структуру даних, що зберігаються, включаючи типи полів і, можливо, їх максимальний розмір, значення за замовчуванням, параметри списку вибору, текст довідки для документації, текст міток для форм і т.п. Після того, як вибрано базу даних, нам не потрібно безпосередньо працювати з нею. Просто пишемо свою структуру моделі та код, а Django робить всю основну роботу, пов'язану з базою даних.

Визначимо перелік об'єктів, які мають вплив на інформаційну систему, і встановимо між ними зв'язки:

- користувачі;
- пацієнти;
- лікарі;
- адміністратори;
- медична карта;
-записи у медичній карті;
- аналізи;
- хронічне захворювання;

-щеплення.

Допоміжними є сутності акаунтів (соціальних мереж), email адрес, зв'язку токенів та дозволів користувачів.

Для початку зв'яжемо базу даних із самим Django. Для запуску Postgres використовується Docker контейнер. Визначаємо налаштування в doche-compose.yaml.

Кешування бази даних - широко використовується метод підвищення масштабованості. Розвантажуючи роботу бази даних в інші, більш швидкі сховища, вона також допоможе поліпшити доступність даних. Тому будемо використовувати також Redis разом із Postgres.

Рисунок 1.2 – Налаштування образів Postgres і Redis [7]

Redis - це система зберігання ключових значень, яка працює в оперативній пам'яті, вона схожа на «легку базу даних», і оскільки вона працює на рівні пам'яті RAM, вона на порядок швидше, ніж читання/запис у PostgreSQL або будь-яку іншу традиційну реляційну базу даних . Redis - це так звана база даних NoSQL, така як Mongo та багато інших. Він не може безпосередньо замінити PostgreSQL, все одно потрібне постійне сховище, але воно працює разом із реляційними базами даних як альтернативна система зберігання. Можна використовувати Redis, якщо операції введення-виведення почнуть дорожчати, і це відмінно підходить для швидких

обчислень та запитів на основі ключів. Головна перевага – зменшення кількості запитів до PostgreSQL.

Запуск бази конфігурації провадиться через команду docker-compose up postgres redis.

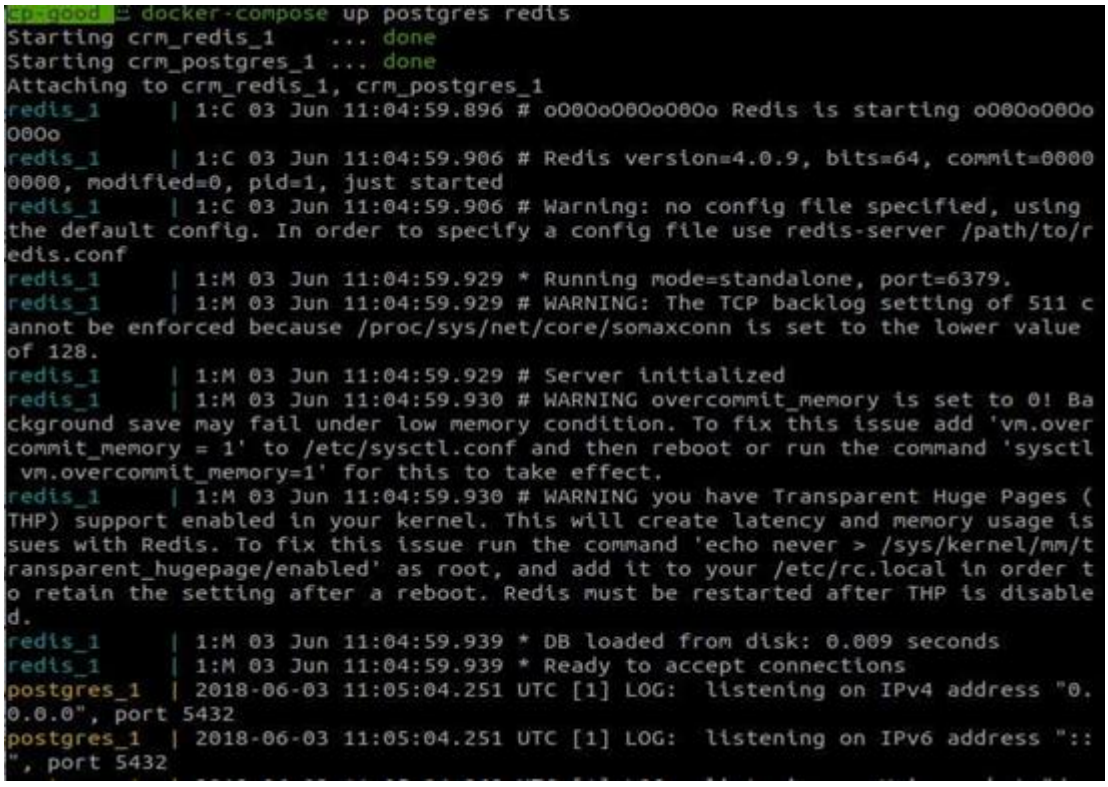

Рисунок 1.3 – Робота образу Postgres і Redis [7]

Тепер необхідно створити таблиці та відносини для бази даних. Було розроблено попередню UML діаграма класів. [4]

Перелічимо основні сутності системи:

- AppUser, базова сутність користувача;
- Patient, розширення користувача, що представляє пацієнта;
- Doctor, розширення користувача, представляє сутність лікаря;
- Medical card, що характеризує сутність медичної карти пацієнта;
- Record, сутність запису у медичній карті;
- Analyze, перелік аналізів пацієнта;
- Inoculation, характеризує записи про звички пацієнта;

- AttachedFile, сутність для різних файлів пацієнта (скан копії документів, аналізів та ін.);

- Working House, сутність, що відображає режим роботи лікарів.

Інші сутності як EmailAddress, Account та інше є частинами зовнішніх бібліотек, що використовується для розширення функцій користувача.

Сутність AppUser є базовою для всіх користувачів. На рисунках 1.4 та 1.5 представлені взаємини між класами.

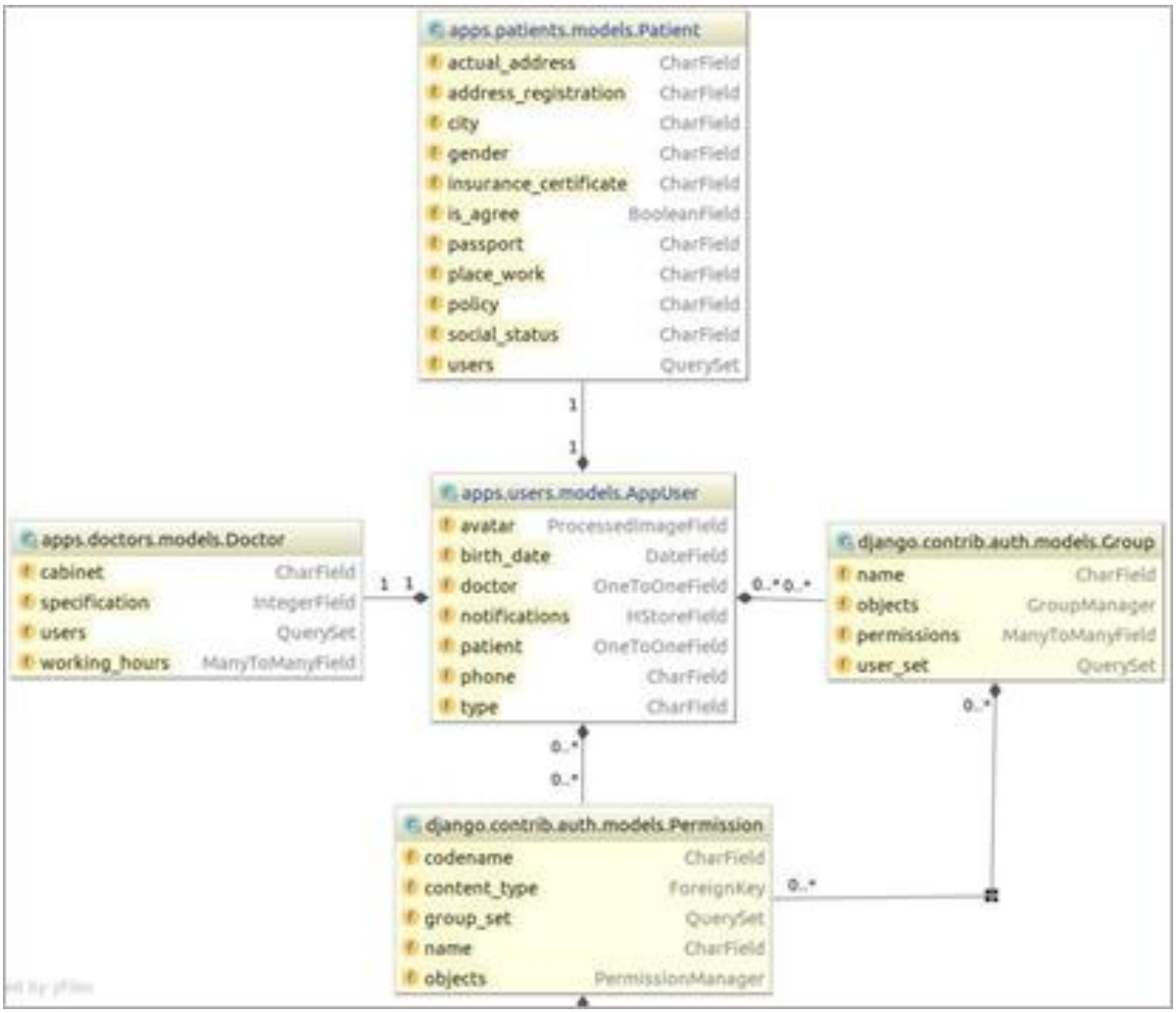

Рисунок 1.4 – Діаграма пов’язаних сутностей AppUser та його розширень [4]

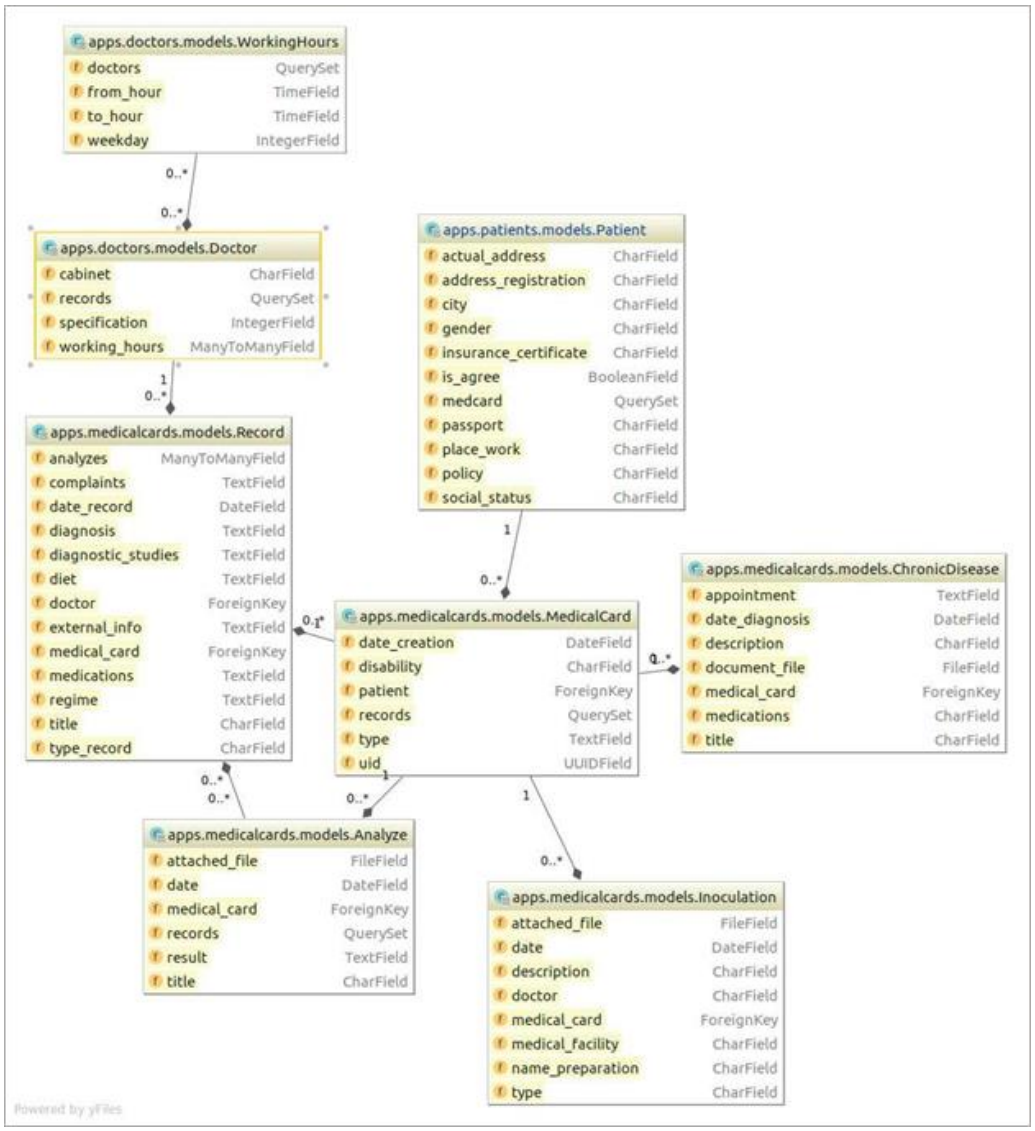

Рисунок 1.5 – Діаграма пов’язаних сутностей із MedicalCards [4]

2 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАСОБУ

Розробка програмних продуктів вимагає не лише створення функціональності, але й перевірки їхньої якості та надійності. Тестування є ключовим етапом у розробці програмного забезпечення, оскільки воно дозволяє виявити потенційні помилки, недоліки та несумісності до його випуску.

Недостатнє тестування може призвести до несподіваних проблем, таких як збої в роботі програми, втрата даних або негативний вплив на користувачів. Тому розробники програмного забезпечення приділяють велику увагу забезпеченню якості своїх продуктів шляхом ретельного тестування.

2.1 Реалізація швидкого створення тестових даних

Щоб додаток було простіше тестувати, і наочніше редагувати зовнішній вигляд, необхідні деякі тестові дані.

Вручну створювати користувачів, пацієнтів – довго та не ефективно, та особливо якщо між сутностями безліч зв'язків, велика кількість полів та інше.[10]

Для спрощення цього завдання в проекті використовуються дві бібліотеки - Factory Boy і Faker. Вони допомагають створити тестові дані швидко, створюючи одночасно всі залежні сутності.

Foreign Key розроблений, щоб добре працювати з різними ORM (Django, Mongo, SQLAlchemy). З Django він працює ідеально - можна навіть налаштувати створення Foreign Key сутностей. Ця бібліотека включає і пакет Faker.

Faker – це пакет Python, який генерує підроблені дані. Незалежно від того, чи потрібно завантажувати базу даних, створювати гарні XML-документи, заповнювати свою наполегливість, проводити стрес-тестування або анонімізувати дані, взяті з виробничої служби.

На рисунку 2.1 представлений клас створення сутності медичної карти.

```
class MedicalCardFactory(factory.django.DjangoModelFactory):
    class Meta:
        model = MedicalCard

    uid = factory.Faker('uuid4')
    date_creation = fake_factory.date_between(start_day, today)
    disability = fake_factory.sentence(nb_words=4,
                                       variable_nb_words=True,
                                       ext_word_list=None)
    type = fake_factory.word()
    patient = factory.SubFactory(PatientFactory)

    record = factory.RelatedFactory(RecordFactory, 'medical_card')
    inoculation = factory.RelatedFactory(InoculationFactory, 'medical_card')
    analyze = factory.RelatedFactory(AnalyzeFactory, 'medical_card')
    chronic_disease = factory.RelatedFactory(ChronicDiseaseFactory,
                                             'medical_card')
```

Рисунок 2.1 – Створення об’єкту моделі медичної карти

Варто звернути увагу, що для всіх зв'язаних сутностей (Аналізи, Записи, Хронічних захворювань, Щеплення, Пацієнт) є окремі фабрики. Тобто медична карта створюється вже з потрібними об'єктами та зв'язками. На рисунку 2.2 подано клас створення об'єкта пацієнта.

```
import factory
import faker

from apps.patients.models import Patient

fake_factory = faker.Faker()


class PatientFactory(factory.django.DjangoModelFactory):

    class Meta:
        model = Patient

    actual_address = factory.Faker('address')
    address_registration = factory.Faker('address')
    city = factory.Faker('city')
    gender = fake_factory.random_element(['M', 'F'])
    policy = fake_factory.uuid4()
    passport = fake_factory.credit_card_number()
    insurance_certificate = fake_factory.credit_card_full()
    place_work = fake_factory.company()
    social_status = fake_factory.random_element(
        ['Student', 'worker', 'unemployed', 'pensioner'])
    user = factory.RelatedFactory('apps.users.factories.UserFactory',
                                  'patient', type='patient')
```

Рисунок 2.2 – Створення об’єкту моделі Пацієнт

Разом із пацієнтами створюється об'єкт користувача з потрібним. Так само створюється і модель Лікар.

Таким чином, в консолі одним рядком *"m = MedicalCardFactory()"* можна створити скільки завгодно об'єктів пацієнтів, медичних карт та інше. Це підвищує ефективність та якість тестування програми.

2.2 Реалізація входу та виходу із системи

Система аутентифікації Django обробляє як аутентифікацію, і авторизацію. Коротко, аутентифікація перевіряє, чи користувач тим, ким він є, а авторизація визначає, що дозволено для аутентифікованого користувача. Тут термін «аутентифікація» використовується для позначення обох задач REST framework (з модуля django-rest-auth). Також надає кілька аутентифікаційних схем та дозволяє вам використовувати свої схеми. [11]

Система аутифікації в Django складається з:

- Користувачів.

- Роздільна здатність: прапори (yes / no), що вказують, чи може користувач виконувати певне завдання.

- Груп. Загальний спосіб застосування міток та дозволів для кількох користувачів.

- Конфігурована система хешування паролів.

- Форм та засобів перегляду для входу в систему користувачів або обмеження вмісту.

Аутентифікація завжди запускається на початку view, до того як відбуватиметься перевірка дозволів і обмежень, і до того як почнеться виконуватися інший код. На рисунках 2.3 та 2.4 представлені форми входу та реєстрації у додатку.

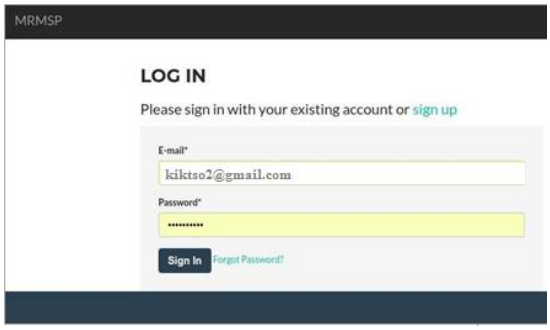

Рисунок 2.3 – Форма входу

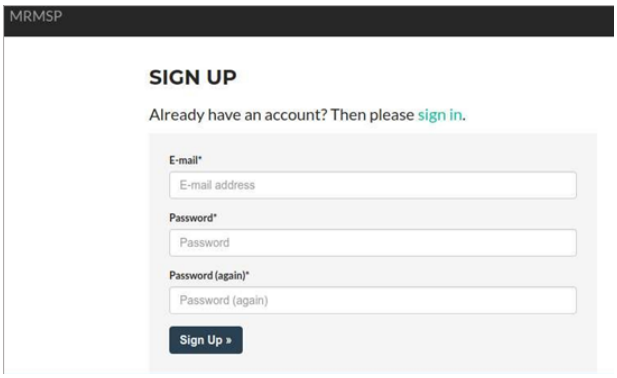

Рисунок 2.4 – Форма реєстрації

Після запуску стартової сторінки користувач може увійти до системи.

При реєстрації користувач спочатку проходить верифікацію адреси. Потім під час першого входу систему заповнюються додаткові дані.

2.3 Реалізація перегляду та редагування користувача

Однією з найзручніших частин Django є інтерфейс автоматичного адміністрування. Він зчитує метадані з моделей, щоб забезпечити швидкий, орієнтований модель інтерфейс. Довірені користувачі можуть керувати вмістом на сайті.

Сайт адміністратора включений за замовчуванням у кожному згенерованому проекті Django.

Після створення моделей ми тепер готові додати їх реєстрацію з допомогою сайту Admin. Для цього потрібно визначати спеціальні класи та реєструвати їх.

Реалізовано панель адміністратора з управління користувачами. Вид загального списку представлений рисунку 2.5.

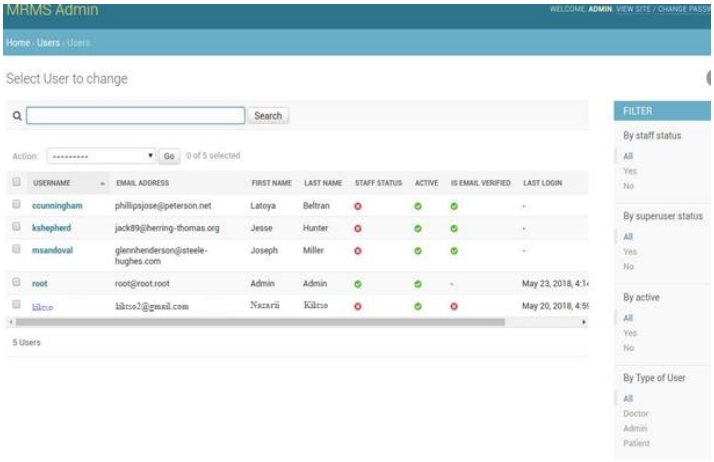

Рисунок 2.5 – Список користувачів в Django Admin

Сторінка редагування користувача представлена на рисунку 2.6.

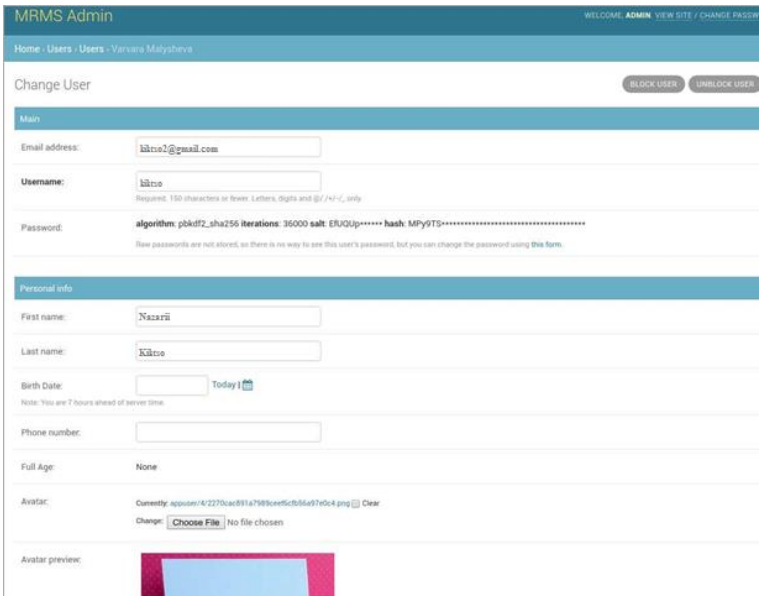

Рисунок 2.6 – Сторінка редагування

Для реалізації функціональності використовується модуль admin у пакеті Django. Можна самостійно налаштувати, які поля відображати, якими фільтрувати тощо.

Також було впроваджено дві додаткові кнопки – блокування та розблокування користувача.

Клас ModelAdmin є представленням моделі в інтерфейсі адміністратора.

Користувача може редагувати лише адміністратор необхідними правами.

2.4 Реалізація інтерфейсу медичної картки

Використовується модуль Django tabbed admin – проста бібліотека для простого додавання вкладок до адмін-форм.

На рисунках 2.7-2.10 представлена панель керування медичною картою пацієнта.

У кожному блоці можна додати / видалити сутності (наприклад, аналізи та інше).

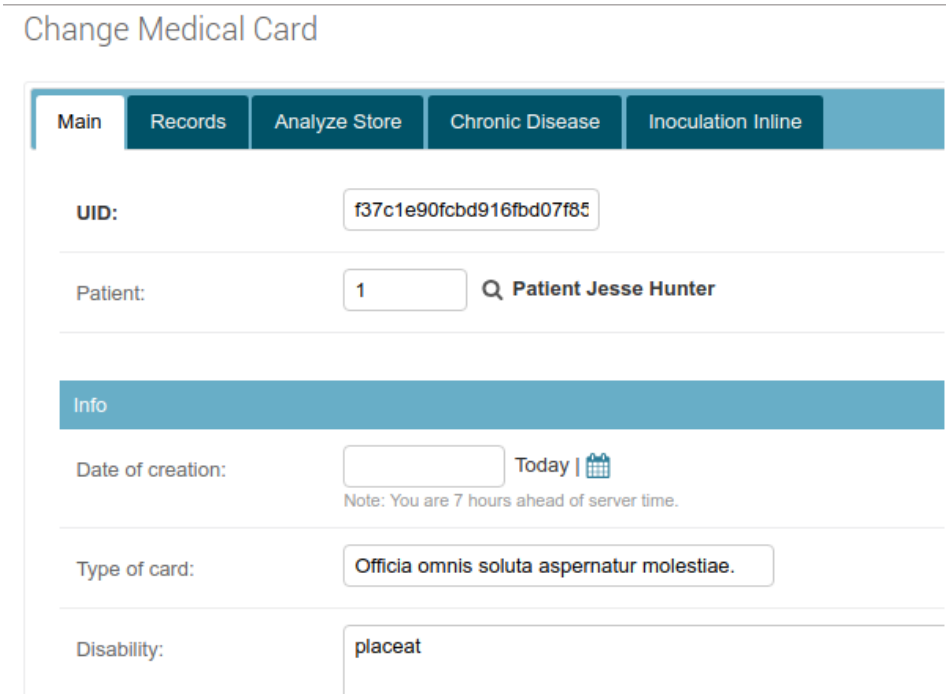

Рисунок 2.7 – Загальний вигляд екрану редагувння медичної карти

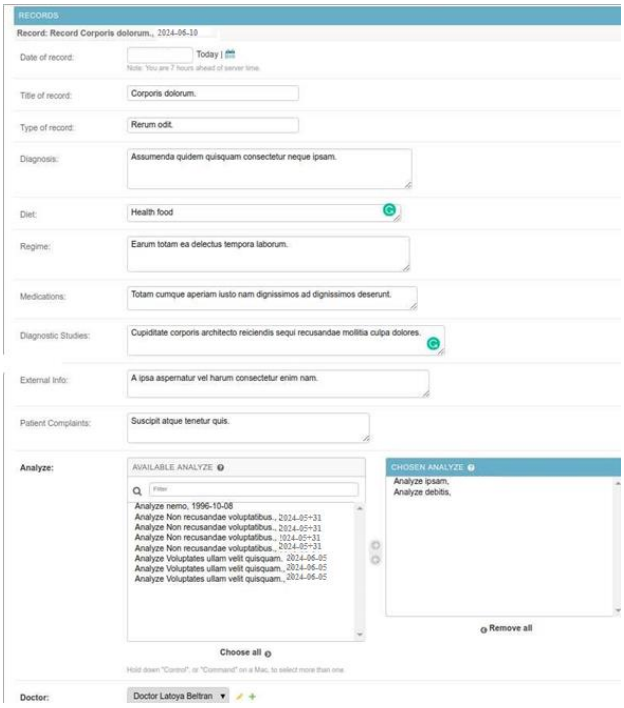

Рисунок 2.8 – Екран редагування записів у медичній карті

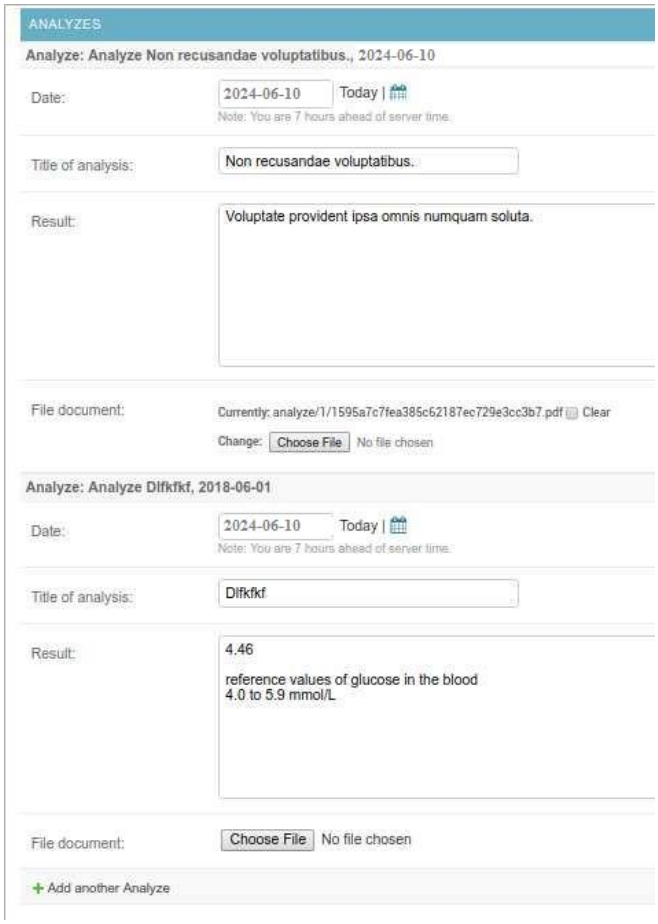

Рисунок 2.9 – Перегляд аналізів

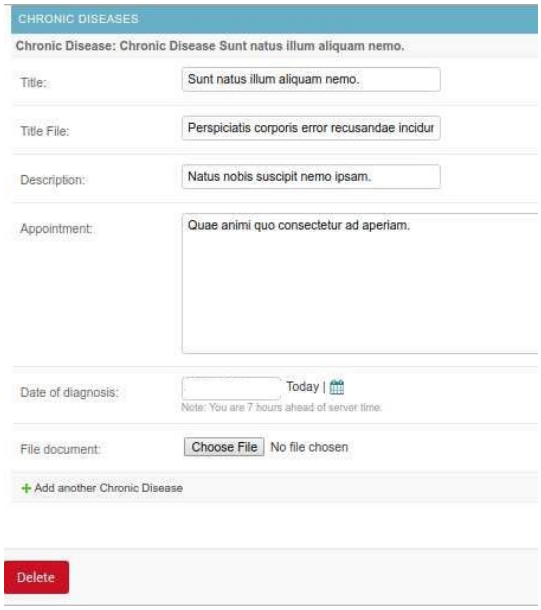

Рисунок 2.10 – Екран перегляду хронічних захворювань

2.5 Реалізація перегляду та редагування даних користувачем

Реалізовано сторінку обновлення полів користувача (див. рис. 2.11).

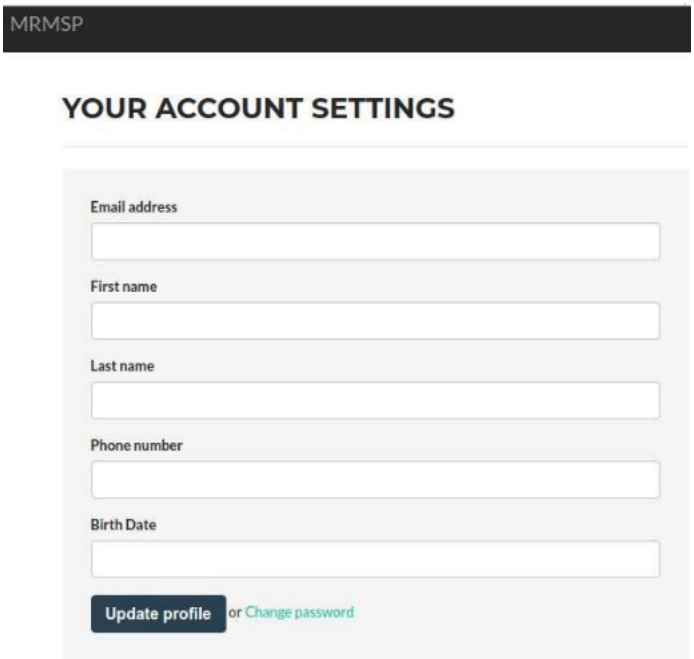

Рисунок 2.11 – Редагування профілю

Також створено форму редагування даних користувача як пацієнта.

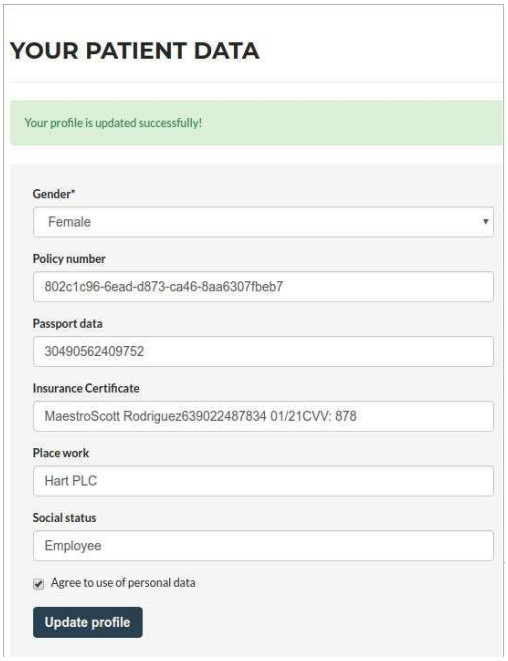

Рисунок 2.12– Редагування даних пацієнта

Для коректної роботи таблиць та пошуку було реалізовано точку API для роботи зі списком об'єктів.

Цей інтерфейс був реалізований за допомогою Django REST framework – бібліотека призначена для створення складних API, що настроюються, і веб сервісів. Були налаштовані види, і типи полів, дозволи на відповідь JSON.

Для рендерингу сторінки використовується таблиця завантаження від фреймворку Boostrap Table.

```python
from rest_framework import viewsets
from rest_framework import serializers
from rest_framework.permissions import AllowAny

from apps.doctors.models import Doctor


class DoctorSerializer(serializers.ModelSerializer):
    working_hours = serializers.CharField(source='all_working_hours')
    specification_string = serializers.CharField(source='str_specification')
    name_doctor = serializers.CharField(source='name')
    class Meta:
        model = Doctor
        fields = '__all__'


class DoctorViewSet(viewsets.ModelViewSet):
    """
    A simple ViewSet for viewing doctors.
    """
    queryset = Doctor.objects.all()
    serializer_class = DoctorSerializer
    permission_classes = [AllowAny]
```

Рисунок 2.13 – Реалізація списку лікарів

```
{% extends "base.html" %}
{% load crispy_forms_tags i18n %}
{% block content %}

    <div class="container">
        <div class="row">
            <h1>Schedule of doctors</h1>
            <hr/>
            <div class="col">
                <table data-toggle="table"
                       data-url="/api/v1/doctors/"
                       data-search-time-out="100"
                       data-id-field="id">
                    <thead>
                    <tr>
                        <th data-field="name_doctor">Doctor Name</th>
                        <th data-field="specification_string">Specification</th>
                        <th data-field="working_hours">Working Hours</th>
                        <th data-field="cabinet">Cabinet</th>
                    </tr>
                    </thead>
                </table>
            </div>
        </div>
    </div>
{% endblock content %}

{% block footer %}
    {% include 'includes/footer.html' with footer_style="navbar-fixed-bottom" %}
{% endblock footer %}
```

Рисунок 2.14 – Шаблон для відображення таблиці розкаладу лікарів

Django REST framework надає засоби для взаємодії із створеним API. Тестувати та перевіряти інтерфейс можна у браузері.

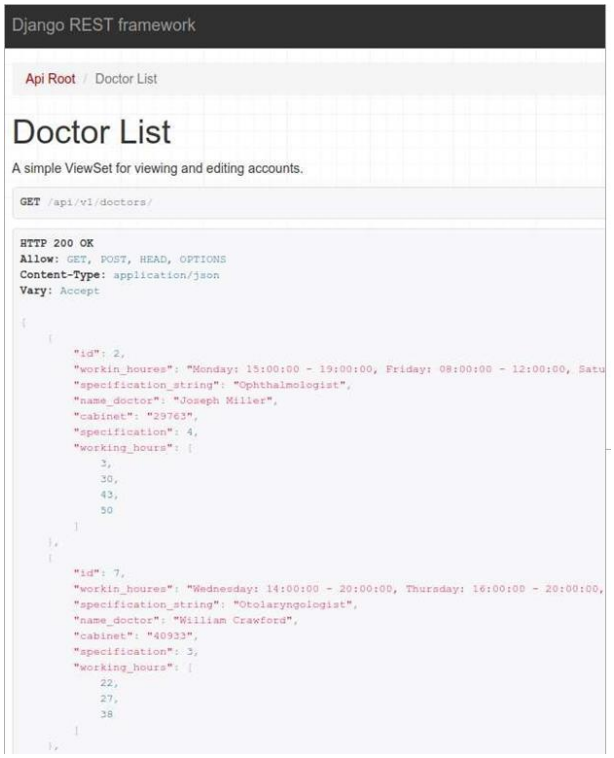

Рисунок 2.15 – Вид JSON відповіді при запиті списку лікарів

2.6 Реалізація сторінки медичної картки користувача

Розроблено інтерфейс перегляду користувачем особистих даних.

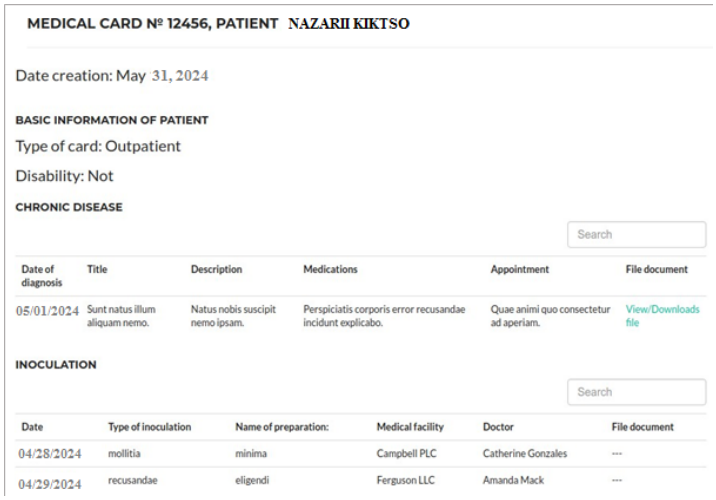

Рисунок 2.16 – Сторінка перегляду даних медичної карти

Користувач може переглядати всі доступні дані, проводити пошук. будована пагінація сторінками. Також користувач має можливість скачати необхідні документи.

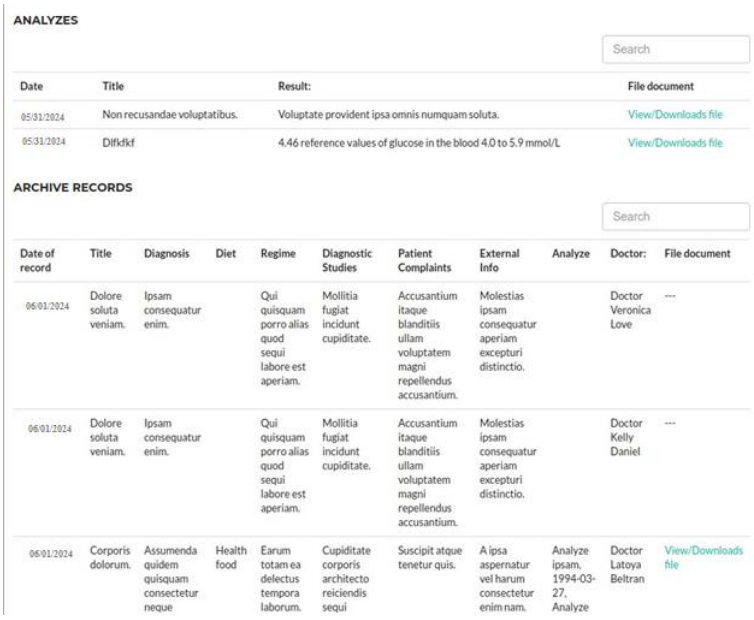

Рисунок 2.17 – Таблиці аналізів та записів прийому в карті пацієнта

3 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

3.1 Порядок надання домедичної допомоги постраждалим при раптовій зупинці серця.

Домедична допомога у разі зупинки серця є надзвичайно важливим заходом, адже за 5-6 хвилин без кровопостачання кора головного мозку починає зазнавати на собі дію процесів, що мають незворотній характер.

Навички першої допомоги – критична необхідність для кожної людини, особливо в умовах війни. Сьогодні це без перебільшень може врятувати життя.

Послідовність дій при наданні домедичної допомоги дорослим при раптовій зупинці кровообігу за відсутності автоматичного зовнішнього дефібрилятора:

1) перед наданням допомоги необхідно переконатися у відсутності небезпеки та за її відсутності перейти до наступного кроку;

2) визначити наявність свідомості – обережно потрясти дорослого за плече та голосно звернутися до нього, наприклад, «З Вами все гаразд? Вам потрібна допомога?»

3) якщо дорослий реагує:

а) залишити його у попередньому положенні, якщо йому нічого не загрожує;

б) з’ясувати характер події, що сталася;

в) здійснити виклик екстреної медичної допомоги та дотримуватись вказівок диспетчера прийому виклику;

г) за необхідності надати дорослому зручного положення;

ґ) забезпечити нагляд за дорослим до приїзду бригади екстреної (швидкої) медичної допомоги;

4) якщо дорослий не реагує:

а) звернутися до осіб, які поряд, за допомогою. Якщо випадкових свідків декілька, слід звертатися до конкретної особи;

б) якщо дорослий лежить на животі, повернути його на спину та відновити прохідність дихальних шляхів. Якщо стан дорослого пов’язаний з отриманням травми, наприклад падіння з висоти, вважати, що у нього є травма в шийному

відділі хребта. В такому випадку слід максимально обмежити рухи в шийному відділі хребта;

в) відновити прохідність дихальних шляхів, визначити наявність дихання за допомогою прийому: «чути, бачити, відчувати». Наявність дихання визначати до 10 секунд. Якщо виникли сумніви чи є дихання, або воно ненормальне, вважати, що дихання відсутнє;

5) якщо дорослий дихає нормально, при відсутності свідомості слід перевести його у стабільне бокове положення, здійснити виклик екстреної медичної допомоги та перевіряти кожні 3-5 хвилин дихання до моменту приїзду бригади екстреної (швидкої) медичної допомоги. Якщо є настороженість щодо травми потрібно уникати повороту в стабільне бокове положення, а забезпечити прохідність дихальних шляхів методом висування нижньої щелепи до приїзду бригади екстреної (швидкої) медичної допомоги;

6) якщо дихання відсутнє:

а) здійснити виклик екстреної медичної допомоги та дотримуватись вказівок диспетчера прийому виклику. Якщо є інші випадкові свідки, слід сказати їм здійснити виклик екстреної медичної допомоги та негайно перейти до наступного кроку. При наявності гучного зв’язку на телефоні слід здійснювати виклик екстреної медичної допомоги та одночасно проводити серцево-легеневу реанімацію;

б) розпочати проведення серцево-легеневої реанімації: виконати 30 натискань на середину грудної клітки глибиною не менше 5 см (не більше 6 см), з частотою 100 натискань (не більше 120) за хвилину; виконати 2 вдихи з використанням маски-клапану, дихальної маски тощо. При відсутності захисних засобів можна не виконувати штучне дихання, а проводити тільки натискання на грудну клітку. Виконання двох штучних вдихів повинно тривати не більше 5 секунд; після двох вдихів продовжити натискання на грудну клітку відповідно до наведених рекомендацій у цьому підпункті; не слід переривати натиснення на грудну клітку дорослому більше ніж на 10 секунд;

7) змінювати особу, що проводить натискання на грудну клітку, кожні 2 хвилини.

8) припинити проведення серцево-легеневої реанімації до прибуття бригадиї екстреної (швидкої) медичної допомоги за наступних умов:

• при появі у дорослого явних ознак життя; відновлення самостійного нормального дихання, координованої рухової активності, відкривання очей; виникненні загрози життю рятівнику та/або дорослому;

• неможливості проведення серцево-легеневої реанімації внаслідок значного фізичного виснаження.

Послідовність дій при наданні домедичної допомоги дорослим при раптовій зупинці кровообігу та наявності автоматичного зовнішнього дефібрилятора:

1) перед наданням допомоги переконатися у відсутності небезпеки та за її відсутності перейти до наступного кроку;

2) визначити наявність свідомості – обережно потрясти дорослого за плече та голосно звернутися до нього, наприклад, «З Вами все гаразд? Вам потрібна допомога?»;

3) якщо дорослий не реагує:

а) звернутися до осіб, які поряд, за допомогою. Якщо випадкових свідків декілька, слід звертатися до конкретної особи;

б) якщо дорослий лежить на животі, повернути його на спину та відновити прохідність дихальних шляхів. Якщо стан дорослого пов’язаний з отриманням травми, наприклад падіння з висоти, вважати, що у нього є травма в шийному відділі хребта. В такому випадку слід максимально обмежити рухи в шийному відділі хребта;

в) відновити прохідність дихальних шляхів, визначити наявність дихання за допомогою прийому: «чути, бачити, відчувати». Наявність дихання визначати до 10 секунд. Якщо виникли сумніви, чи є дихання, або воно ненормальне, вважати, що дихання відсутнє;

г) здійснити виклик екстреної медичної допомоги та дотримуватись вказівок диспетчера прийому виклику. Якщо є інші випадкові свідки, слід сказати їм здійснити виклик екстреної медичної допомоги та негайно перейти до наступного кроку. При наявності гучного зв’язку на телефоні слід здійснювати виклик екстреної медичної допомоги та одночасно проводити серцево-легеневу

реанімацію. Якщо рятувальник один, і не відомо, що автоматичний зовнішній дефібрилятор знаходиться у безпосередній близькості від місця події, не слід залишати дорослого та шукати автоматичний зовнішній дефібрилятор – в такому випадку слід проводити серцево-легеневу реанімацію як передбачено у пункті 3 цього Порядку;

г) розпочати проведення серцево-легеневої реанімації: виконати 30 натискань на середину грудної клітки глибиною не менше 5 см (не більше 6 см), з частотою 100 натискань (не більше 120) за хвилину; виконати 2 вдихи рекомендовано з використанням маски-клапану, дихальної маски тощо. При відсутності захисних засобів можна не виконувати штучне дихання, а проводити тільки натискання на грудну клітку. Виконання двох штучних вдихів повинно тривати не більше 5 секунд; після двох вдихів продовжити натискання на грудну клітку відповідно до наведених рекомендацій у цьому підпункті; не слід переривати натискання на грудну клітку дорослому більше ніж на 10 секунд;

4) змінювати особу, що проводить натискання на грудну клітку, кожні 2 хвилини. У випадку якщо особа, яка проводить натискання на грудну клітку, відчуває виснаження, заміну слід виконати раніше, ніж через 2 хвилини;

5) як тільки автоматичний зовнішній дефібрилятор наявний на місці події, слід негайно: увімкнути пристрій та чітко дотримуватись голосових вказівок; якщо осіб які надають допомогу декілька, вмикання автоматичного зовнішнього дефібрилятора та приклеювання електродів до грудної клітки дорослого слід одночасно з проведенням компресій на грудну клітку; у випадку необхідності проведення дефібриляції, прослідкувати, щоб ніхто не торкався дорослого; після проведення дефібриляції слід негайно розпочати натискання на грудну клітку;

6) припинити проведення серцево-легеневої реанімації до прибуття бригади екстреної (швидкої) медичної допомоги за наступних умов:

• при появі у дорослого явних ознак життя; відновлення самостійного дихання, координованої рухової активності; виникненні загрози життю рятівнику та/або дорослому;

У випадку появи явних ознак життя до приїзду бригади екстреної (швидкої) медичної допомоги, електроди від автоматичного зовнішнього дефібрилятора слід залишити на грудній клітці дорослого.

3.2 Планування робіт щодо охорони праці

У плануванні робіт з охорони праці як вихідні дані використовують:

· план економічного і соціального розвитку підприємства;

· план організаційно-технічних, технологічних та інших заходів, направлених на підвищення ефективності, пошук резервів підприємства;

· матеріали паспортизації цехів, дільниць, робочих місць на відповідність їх вимогам охорони праці; а також атестації робочих місць за умовами праці;

· комплексні заходи щодо досягнення встановлених нормативів безпеки, гігієни праці і виробничої санітарії;

· матеріали розслідувань нещасних випадків, професійних захворювань та аварій;

· приписи органів державного нагляду, служб охорони праці;

· рекомендації комісії з питань охорони праці підприємства; пропозиції уповноважених трудового колективу з питань охорони праці;

· результати адміністративно-громадського контролю стану охорони праці;

· постанови профспілкового комітету, рішення зборів (конференцій) трудового колективу з питань охорони праці;

· відповідні накази адміністрації, документи вищестоящих господарських і профспілкових органів.

Важливе значення у системі планування робіт щодо забезпечення безпеки праці на підприємствах має розробка розділу «Охорона праці» колективного договору (угоди, трудового договору), що укладається між власником і трудовим колективом, за рішенням якого його інтереси може представляти також і профспілка. Одним з найважливіших розділів колективного договору є інженерно-технічні заходи задля досягнення нормативівбезпеки, гігієни праці і

виробничого середовища, підвищення наявного рівня охорони праці. До цільових заходів належать, наприклад, розробка, виготовлення і установка нових, ефективніших інженерно-технічних засобів охорони праці, огорож, засобів сигналізації і контролю, запобіжних пристроїв тощо.

Заходи з охорони праці мають бути забезпечені проектно-кошторисно-конструкторською та іншою технічною документацією, фінансуванням і матеріальними ресурсами.

Згідно зі статтею 19 закону України «Про охорону праці», фінансування охорони праці на підприємстві здійснюється власником. На підприємствах, в галузях і на державному рівні створюються фонди охорони праці. Такі ж фонди створюються органами місцевого і регіонального самоврядування для потреб регіону. В державний, галузеві та регіональні фонди охорони праці направляються, разом із коштами державного і місцевого бюджетів, відрахування підприємств (в розмірі не менше 0,5% від об'єму реалізації продукції, виконаних робіт, наданих послуг, а для бюджетних підприємств – не менше 0,2% від фонду оплати праці) та іншими надходженнями, кошти, отримані від застосування органами державного нагляду штрафних санкцій до підприємств, а також кошти від штрафування фізичних осіб, винних у порушеннях вимог охорони паці.

На підприємствах фінансування заходів з охорони праці може здійснюватись, зазвичай за рахунок:

· виробничих витрат (матеріальні витрати на удосконалення технології та організації виробництва, підтримку основних виробничих фондів у робочому стані, утримання засобів колективного захисту та ін.);

· амортизації основних фондів;

· капітальних вкладень;

· банківського кредиту;

· кошторису затрат бюджетних організацій і установ;

· фонду охорони праці.

Засоби фонду охорони праці підприємства повинні використовуватися тільки на виконання комплексних заходів, спрямованих на досягнення встановлених нормативів з охорони праці, подальше підвищення рівня охорони

праці в умовах діючого виробництва. До цих заходів, які є основою для розробки колективного договору, належать:

· атестація робочих місць на відповідність їх вимогам нормативних актів про охорону праці;

· навчання працівників з питань охорони праці;

· забезпечення працівників спецодягом, спецвзуттям та іншими засобами індивідуального захисту;

· проведення обов'язкових медичних оглядів працівників;

· проведення експертизи технічного стану будівель і споруд;

· приведення рівня шуму, вібрації, ультразвуку, іонізуючих та інших шкідливих випромінювань на робочих місцях у відповідність вимогам чинних нормативних актів;

· розробка, виготовлення і монтаж нових або реконструкція існуючих вентиляційних систем, аспіраційних, порохо- і газоуловлювальних пристроїв для приведення у відповідність вимогам чинних нормативних актів про охорону праці та їх паспортизація тощо.

Першочерговість виконання заходів встановлюється на основі атестації робочих місць на відповідність їх вимогам нормативних актів з охорони праці, аналізу причин нещасних випадків на виробництві і професійних захворювань з урахуванням їх економічної і соціальної значимості.

Засоби фонду охорони праці підприємства не повинні витрачатися на ремонтні та інші роботи, пов'язані з підтриманням в належному технічному стані основних фондів (включаючи інженерно-технічні засоби безпеки, засоби колективного захисту працівників), на надання пільг і компенсацій працівникам, на природоохоронні заходи.

Засоби галузевих і регіональних фондів охорони праці підприємствам направляються згідно з затвердженими кошторисами витрат на фінансування заходів, передбачених відповідно галузевими і регіональними програмами покращення стану безпеки, гігієни праці та виробничої санітарії.

До заходів, які здійснюються за рахунок коштів галузевого фонду охорони праці, належать, наприклад:

· створення і впровадження автоматизованих систем попередження аварій, пожеж, нещасних випадків на виробництві, а також систем управління охороною праці;

· розробка і виконання програм метрологічного, кадрового і організаційного забезпечення атестації робочих місць;

· нормативно-правове забезпечення охорони праці;

· проведення і впровадження у виробництво науково-дослідних, проектно-конструкторських робіт з безпеки і гігієни праці.

За рахунок регіонального фонду охорони праці виконуються, наприклад, такі заходи :

· приведення у відповідність з вимогами нормативних актів приміщень для зберігання отруйних, пожежо- і вибухонебезпечних речовин;

· утилізація небезпечних і шкідливих відходів виробництва;

· забезпечення безпечної експлуатації об’єктів з постійним перебуванням великої кількості людей (магазинів, навчальних закладів та ін.).

Керівництво державним фондом охорони праці здійснює Державний комітету країни з нагляду за охороною праці (Держгірпромнагляд). Кошти цього фонду відповідно до кошторису, затвердженого Держгірпромнаглядом за узгодженням з Мінфіном, направляються на фінансування заходів комплексного вирішення завдань з охорони праці, передбачених національною програмою покращення стану безпеки, гігієни праці і виробничого середовища, проведення науково-дослідних і проектно-конструкторських робіт в межах цієї програми, сприяння становленню і розвитку спеціалізованих підприємств і виробництв, науково-технічних центрів і експертних груп та здійснення цих заходів.

Таким чином, на Україні створена і функціонує єдина система планування і фінансування заходів з охорони праці: на підприємствах і в галузях, на регіональному і державному рівнях.

ВИСНОВОК

В результаті виконання бакалаврської роботи було розроблено ІС організації роботи медичного реєстратора, в частині ведення електронних медичних карток пацієнтів. У процесі проектування та розробки було вирішено всі завдання та досягнуто всі поставлені цілі.

Розроблена надалі програмна система дозволить переглядати зведену інформацію про лікарів (відображення списку лікарів з назвами їх спеціальностей) та пацієнтів (відображення списку пацієнтів та номерів їх карток), видаляти та змінювати дані. Передбачено інтерфейс для пацієнтів – вони зможуть переглядати свої особисті дані та записи у своїй медичній карті. Також пацієнт має доступ до розкладу роботи лікарів.

У ході виконання випускної кваліфікаційної роботи було повністю виконано такі завдання:

- обґрунтовано актуальність завдання;
- проведено аналіз аналогів медичних систем керування медичними картами пацієнтів;
- визначено інструментальні засоби та початкові вимоги;
- розроблено UML діаграма класів;
- створені моделі БД за допомогою ORM Django;
- створено БД системи, застосовані міграції;
- реалізовано адміністративний інтерфейс управління користувачами, пацієнтами та медичними картами у Django Admin;
- реалізовано стартову сторінку програми;
- впроваджено сторінку перегляду розкладу роботи лікарів;
- впроваджено онлайн-перегляд пацієнтами медичної карти та результатів аналізів;
- створено тестові дані та проведено тестування;

Впровадження такої програмної системи може значно скоротити час обробки інформації, здійснити швидкий пошук необхідної інформації,

автоматично формувати звіти, дозволяти зберігати великі обсяги інформації та уникнути втрати інформації.

СПИСОК ЛІТЕРАТУРИ

1. Веб-додаток і його характеристики [Електронний ресурс] https://www.centum-d.com/uk/veb-dodatok-yogo-harakteristiki
2. Introduction to DOM elements [Електронний ресурс] https://frontender.info/an-introduction-to-dom-events/
3. Сучасний веброзробник [Електронний ресурс] https://medium.com/nuancesofprogramming
4. EMR [Електронний ресурс] https://www.capterra.com/electronic-medical-records-software/
5. Python Beginning [Електронний ресурс] :https://prjctr.com/course/python-beginning/
6. DjangoSQL [Електронний ресурс] https://github.com/django/django/blob/main/LICENSE
7. PostgreSQL [Електронний ресурс] https://www.postgresql.org/download/
8. Git [Електронний ресурс] https://git-scm.com
9. Front-end і Back-end розробка [Електронний ресурс] https://skillbox.ua/media/code/frontend_i_backend_rozrobka/
10. Different Types of Web Application Development [Електронний ресурс] https://www.clustox.com/6-different-types-of-web-application-development
11. Automation Testing Life Cycle [Електронний ресурс] https://www.lambdatest.com/blog/all-you-have-to-know-about-automation-testing-lifecycle
12. JavaScript unit testing frameworks in 2020: A comparison [Електронний ресурс] https://raygun.com/blog/javascript-unit-testing-frameworks/
13. Best JavaScript Unit Testing Framework and Tools [Електронний ресурс] https://geekflare.com/javascript-unit-testing/
14. QA Wolf [Електронний ресурс] https://docs.qawolf.com/docs/
15. Slack API [Електронний ресурс] https://api.slack.com/web

16. Методичні вказівки до виконання дипломної роботи освітнього рівня ―бакалавр студентами усіх форм навчання для напряму підготовки 121 – Інженерія програмного забезпечення‖ / Укладачі : Петрик М.Р., Михалик Д.М., Кінах Я.І., Гладьо С.В., Цуприк Г.Б. – Тернопіль : Вид-во ТНТУ імені Івана Пулюя, 2016 – 28 с.

17. М.Р. Петрик, Д.М. Михалик, О.Ю. Петрик, Г.Б. Цуприк. Методичні вказівки до виконання атестаційної роботи магістра за спеціальністю 121 – “Інженерія програмного забезпечення” для усіх форм навчання [Текст] – Тернопіль : Тернопільський національний технічний університет імені Івана Пулюя – 2020 – 27 с.

18. Бойко І.В., М.Р. Петрик, Г.Б. Цуприк. Інформаційні технології видобутку даних (Data mining, високопродуктивні обчислення у складних системах): навчальний посібник. Тернопіль: : ТНТУ 2020 – 62 с.

19. Бойко І.В., М.Р. Петрик, Г.Б. Цуприк. Моделювання та видобуток даних (висопродуктивні обчислення у великих алгебраїчних та числових системах, комбінаторному аналізі): навчальний посібник. Тернопіль: : ТНТУ 2019 – 62 с.

20. Нога В.В., Цуприк Г.Б. Інформаційні технології при створенні автоматизованих систем для задач збору та збереження інформації. Збірник тез доповідей VII Міжнародної науково-технічної конференції молодих учених та студентів. Актуальні задачі сучасних технологій – Тернопіль 28-29 листопада 2018. — Т. : ТНТУ, 2018. — Том 2. — С. 131.

21. І.В. Бойко, М.Р. Петрик, Г.Б. Цуприк. Дискретні структури (Алгебраїчні та числові системи, комбінаторний аналіз). Навчально-методичний посібник для студентів спеціальності 121 «Інженерія програмного забезпечення», аспірантів та викладачів вищих навчальних закладів. Конспект лекцій : Навчальний посібник / І.В. Бойко, М.Р. Петрик, Г.Б. Цуприк – Тернопіль : ТНТУ , 2017. – 64 с.

ДОДАТКИ

ДОДАТОК А

Тест план та результати тестування

На лістингах представлено реалізацію модульних тестів методів створення та редагування даних відповідно.

клас Cwork

```
# include
# include
# include "Worker. h "CWork
{::: vector pr; WorkBegin;
WorkEnd;
Work [20];
WorkPlace [25];
Reason [40];:
getWorkBegin (void);
setWorkBegin (int AWorkBegin );
getWorkEnd (void); setWorkEnd (int AWorkEnd);
* getWork (void);
setWork (char * AWork);
* getWorkPlace (void);
setWorkPlace (char * APlace);
* getReason (void);
setReason (char * AReason );
AddWorkerIntoWork (CWorker * g);
RemoveWorker (CWorker * g);
CountWorkerInWork ();
WriteV ();
WriteFV (std :: ofstream & t);
ReadFV (std :: ifstream & t);
Read ();
Write ();
ReadF (std :: ifstream & t);
WriteF (std :: ofstream & t);
();
(CWork & As);
(int AWorkBegin, int AWorkEnd, char * AWork, char * AWorkPlace , char * AReason);
};
```

клас Cworker

```
# ifndef _PEO__
# define _PEO__
# include
# include CWorker
{: PersonID; * Family; Birth [12];: getPersonID (void); setPersonID (int APersonID); * getFamily
(void); setFamily (char * AFamily); * getBirth (void) ; setBirth (char * ABirth); Read (); Write ();
ReadF (std :: ifstream & f); WriteF (std :: ofstream & f); (); (CWorker & As); (int APersonID, char *
AFamily, char * ABirth);
~ CWorker ();
};
# endif
```

клас CDocent

```
# include
# include "Worker. h "CDocent: public CWorker
```

```
{: GodZ; UchSpec [15];: getGodZ (void); setGodZ (int AGodZ); * getUchSpec (void); setUchSpec
(char * AUchSpec); Write (); Read (); WriteF (std :: ofstream & t); ReadF (std :: ifstream & t); ();
(CDocent & As); (int APersonID, char * AFamily, char * ABirth, int AGodZ, char * AUchSpec );
};
```

Файл Work. cpp

```
# include "stdafx. h "
# include
# include "Work. h "
# include "Worker. h "
# include "Docent. h "
# include
# include std :: vector v; std :: vector w; std :: vector v2; CWork :: AddWorkerIntoWork (CWorker *
g)
{. push_back (g);
gn="justify">} CWork :: WriteV ()
{<<"Kollichestvo rabotnikov:" <
{[i] -> Write ();
}
} CWork :: WriteFV (std :: ofstream & t)
{<
{(int n = 0; n
{(v [n] == pr [i]) break;
} <
}
} CWork :: ReadFV (std :: ifstream & t)
{N2, n2;>> N2; (int i = 0; i
{>> n2;. push_back (v [n2]);
}
} CWork :: CountWorkerInWork ()
{pr. size ();
} CWork :: getWorkBegin (void) {WorkBegin;
} CWork :: setWorkBegin (int AWorkBegin) {= AWorkBegin;
} CWork :: getWorkEnd (void) {WorkEnd;
} CWork :: setWorkEnd (int AWorkEnd) {= AWorkEnd;
} * CWork :: getWork (void) {Work;
} CWork :: setWork (char * AWork) {(Work, AWork);
} * CWork :: getWorkPlace (void) {WorkPlace;
} CWork :: setWorkPlace (char * AWorkPlace) {(WorkPlace, AWorkPlace);
} * CWork :: getReason (void) {Reason;
} CWork :: setReason (char * AReason) {(Reason, AReason);
} CWork :: Read ()
{<<"Vvedite WorkBegin" <> WorkBegin; <<"Vvedite WorkEnd" <> WorkEnd ; <<"Vvedite Work"
<> Work; <<"Vvedite WorkPlace" <> WorkPlace; <<"Vvedite Reason "<> Reason; i; command = 1; *
Temp100; (command! = 0)
{<<"Please enter operation." <> command; (command == 1)
{= new CWorker (); -> Read ();. push_back (Temp100); (Temp100);
} if (command == 2)
{<<"Vvedite nomer Workera iz globalnogo spiska";>> i; (v [i]);
}
{<<"Invalid command" <
}
}
} CWork :: Write ()
```

```
{<<"WorkBegin -" <
} CWork :: ReadF (std :: ifstream & t)
{>> WorkBegin>> WorkEnd>> Work>> WorkPlace>> Reason; (t);
} CWork :: WriteF (std :: ofstream & t)
{<
<
} :: CWork ()
{= 1990; = 2000; (Work, "Director"); (WorkPlace, "Lotus");
strcpy (Reason, "Po sobstvennomu jelaniyu");
} :: CWork (CWork & As)
{= As. WorkBegin; = As. WorkEnd; (Work, As. Work); (WorkPlace, As. WorkPlace); (Reason, As.
Reason);
} :: CWork (int AWorkBegin, int AWorkEnd, char * AWork, char * AWorkPlace, char * AReason)
{= AWorkBegin; = AWorkEnd; (Work, AWork); (WorkPlace, AWorkPlace); (Reason, AReason);
}
```

Файл Worker. cpp

```
# include "StdAfx. h "
# include
# include "Worker. h "
# include CWorker :: getPersonID (void) {PersonID;
} CWorker :: setPersonID (int APersonID) {= APersonID;
} * CWorker :: getFamily (void) {Family;
} * CWorker :: getBirth (void) {Birth;
} CWorker :: setBirth (char * ABirth)
{(Birth, ABirth);
} CWorker :: Read ()
{<<"Vvedite PersonID" <> PersonID; <<"Vvedite Family" <> Family ; <<"Vvedite Birth" <> Birth;
} CWorker :: Write ()
{<<"PersonID -" <
} CWorker :: ReadF (std :: ifstream & f)
{>> PersonID>> Family>> Birth;
} CWorker :: WriteF (std :: ofstream & f)
{<
} :: CWorker ()
{= 10; = new char [14]; (Family, "Ivanov"); (Birth, "01.01.10");
} :: CWorker (CWorker & As)
{= As. PersonID; = new char [strlen (As. Family) +1]; (Birth, As. Birth);
} :: CWorker (int APersonID, char * AFamily, char * ABirth)
{= APersonID; = new char [strlen (AFamily) +1]; (Family, AFamily); (Birth, ABirth);
} :: ~ CWorker ()
{[] Family;
} CWorker :: setFamily (char * AFamily)
{(Family! = 0) [] Family; = new char [strlen (AFamily) +1]; (Family, AFamily);
}
```

Файл Docent. cpp

```
# include "stdafx. h "
# include
# include "Docent. h "
# include "Worker. h "
>
# include CDocent :: getGodZ (void)
{GodZ;
} CDocent :: setGodZ (int AGodZ)
```

```
{= AGodZ;
} * CDocent :: getUchSpec (void) {UchSpec;
} CDocent :: setUchSpec (char * AUchSpec) {(UchSpec, AUchSpec);
} :: CDocent (): CWorker ()
{= 1997; (UchSpec, "Docent");
} :: CDocent (int APersonID, char * AFamily, char * ABirth, int AGodZ, char * AUchSpec): CWorker
(APersonID, AFamily, ABirth)
{= AGodZ; (UchSpec, AUchSpec);
} :: CDocent (CDocent & As)
{= As. GodZ; (UchSpec, As. UchSpec);
} CDocent :: Read (void)
{<<"vvedite informazhiy o GodZ and UchSpec" <> GodZ; <<"vvedite UchSpec" <> UchSpec;
} CDocent :: ReadF (std :: ifstream & t)
{:: ReadF (t);>> GodZ>> UchSpec;
} CDocent :: Write (void)
{<
} CDocent :: WriteF (std :: ofstream & t)
{:: WriteF (t); <
}
```

Файл main. cpp

```
# include "stdafx. h "
# include
# include
# include
# include "Work. h "
# include "Worker. h "
# include "Docent. h "
# include ReadFAllCWorker (char * tn, std :: vector & v)
{:: ifstream t (tn); N;>> N; * m; (int i = 0; i
{= new CWorker ();> ReadF (t);. push_back (m);
}. close ();
} WriteFAllCWorker (char * tn, std :: vector v)
{:: ofstream t (tn); <
{[i] -> WriteF (t);
}. close ();
} WriteAllCWorker (std :: vector v)
{<<"kolich Worker =" <
{[i] -> Write ();
}
} ReadFAllCWork (char * tn, std :: vector & w)
{:: ifstream t (tn); N;>> N; * m; (int i = 0; i
{= new CWork ();> ReadF (t);. push_back (m);
}. close ();
} WriteFAllCWork (char * ...
tn, std :: vector w)
{:: ofstream t (tn); <
{[i] -> WriteF (t);
}. close ();
} WriteAllCWork (std :: vector w)
{<<"kolich Workov =" <
{[i] -> Write ();
}
} ReadFAllCDocent (char * tn, std :: vector & v2)
```

```
{:: ifstream t (tn); N;>> N; * m; (int i = 0; i
{= new CDocent ();> ReadF (t);. push_back (m);
}. close ();
} WriteFAllCDocent (char * tn, std :: vector v2)
{:: ofstream t (tn); <
{[i] -> WriteF (t);
}. close ();
} WriteAllCDocent (std :: vector v2)
{<<"kolich specialnostei =" <
{[i] -> Write ();
}
} :: vector v; :: vector w; :: vector v2; main (void)
{("Worker. txt", v); ("Work. txt", w); ("Docent. txt", v2); tn [600]; kod; { < vector "< File "< Display
"< vector "< File "< Display "< vector "< File "< Display "<> kod; (kod)
{1: <> tn; (tn, v);; 2: <> tn; (tn, v);; 3: (v);; 4: * m; = new CWorker;> Read ();. push_back (m);; 5: <>
tn; (tn, w);; 6: << ; endl; <<"Input File name:" <> tn; (tn, w);; 7: (w);; 8: * a; = new CWork; > Read
();. push_back (a);; 9: <> tn; (tn, v2);; 10: << ; endl; <<"Input File name:" <> tn; (tn, v2);; 11: (v2);;
12: * o; = new CDocent; > Read ();. push_back (o);;
}
} while (kod> 0); <<"GOOD BYE";
}
```

Додаток Б. Диск з роботою