

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіоелектроніки
Факультет радіоелектроніки та телекомунікацій
(повне найменування інституту, факультету)

Кафедра радіотехніки та телекомунікацій
(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

_____ магістра _____

(ступінь вищої освіти)

на тему ІНФОРМАЦІЙНА СИСТЕМА ПРИВАТНОЇ
МЕДИЧНОЇ УСТАНОВИ

Виконав: студент 2 курсу, групи РТ-220м

Спеціальності _____

172 «Телекомунікації та радіотехніка»

(код і найменування спеціальності)

Освітня програма (спеціалізація)

«Телемедичні та біомедичні системи»

Комір М.О.

(прізвище та ініціали)

Керівник _____

Морщавка С.В.

(прізвище та ініціали)

Рецензент _____

Фарафонов О.Ю.

(прізвище та ініціали)

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

Інститут, факультет Інститут інформатики та радіoeлектроніки, ФРЕТ
 Кафедра Радіотехніка та телекомунікації.
 Ступінь вищої освіти Магістр
 Спеціальності 172 «Телекомунікації та радіотехніка»
 (код і найменування)
 Освітня програма (спеціалізація) «Телемедичні та біомедичні системи»
 (назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри РТТ

 Морщавка С.В.

« » 20 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Комір Микола Олександрович

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Інформаційна система приватної медичної установи

керівник проєкту (роботи) Морщавка Сергій Володимирович, к.т.н., доцент,
 (прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від «01» листопада 2021 року №411

2. Строк подання студентом проєкту (роботи) 13 грудня 2021 року

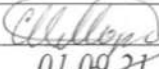
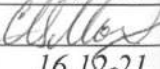
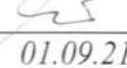
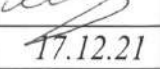
3. Вихідні дані до проєкту (роботи) Розробити інформаційну систему для організації запису пацієнтів до лікарів приватної клініки з можливістю самостійно обрати лікаря, часу та дня відвідування, самостійно відмінити запис до лікаря. Крім того, потрібно забезпечити можливість збереження та аналізу призначень лікарських препаратів лікарями клініки. При розробці виходити з кількості лікарів медичної установи – менше 50, з середнім навантаження 1000 пацієнтів на лікаря.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Належить обрати склад та структуру медичної інформаційної системи, платформу та мову програмування для її реалізації. Детально проаналізувати питання взаємодії інформаційної системи з заявками пацієнтів та з точки зору планування робочого часу лікарів. Розглянути питання економічної доцільності впровадження такої системи та питання охорони праці в галузі.

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

Презентація виконаного проєкту, що відображає структуру розробленої системи, організацію обміну медичною інформацією, взаємодію з пацієнтами, тощо.

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4	Морщавка С.В., доцент	 01.09.21	 16.12.21
5	Севастьянов Р.В., доцент	 01.09.21	 15.12.21
6	Якімцов Ю.В., доцент	 01.09.21	 11.12.21
Нормоконтроль	Мороз Г.В., ст.викладач	 01.09.21	 17.12.21

7. Дата видачі завдання «01» вересня 2021 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Актуальність роботи та постановка задачі на її виконання	3-й тиждень	виконано
2	Аналіз медичної інформаційної інфраструктури	4-й тиждень	виконано
3	Аналіз медичної інформаційної інфраструктури в інших країнах	5-й тиждень	виконано
4	Типи медичної інформаційної інфраструктури	6-й тиждень	виконано
5	Методи планування роботи	7-й тиждень	виконано
6	Дослідження серверної архітектури медичної інформаційної інфраструктури	8-й тиждень	виконано
7	Розробка медичної інформаційної інфраструктури	9-й тиждень	виконано
8	Розробка економічної частини	12-й тиждень	виконано
9	Розробка частини з охорони праці	14-й тиждень	виконано
10	Оформлення пояснювальної записки та презентації	15-й тиждень	виконано
11	Проходження нормконтролю, рецензування, антиплагіату	15-й тиждень	виконано

Студент

Керівник проєкту (роботи)

Комір М.О.
(прізвище та ініціали)

Морщавка С.В.
(прізвище та ініціали)

РЕФЕРАТ

ПЗ: 127 сторінок, 32 рисунки, 20 таблиць, 1 додаток, 27 джерел

Об'єкт дослідження – медична інформаційна інфраструктура, її типи, та функціональні особливості.

Предмет дослідження – процес прототипування медичної інформаційної системи приватної медичної установи.

Метою дипломної роботи є розробка медичної інформаційної системи приватної медичної установи.

Як інструмент роробки викристовується інтегроване програмне середовище. Також використовуються мови програмування високого рівня.

Задачі магістерської роботи:

- виконати дослідження існуючої медичної інформаційної інфраструктури в Україні
- виконати дослідження типів медичної інформаційної інфраструктури, визначити її фнкціональні особливості.
- виконати програмну реалізацію медичної інформаційної інфраструктури.

Виконана розробка має значну практичну цінність, адже дозволяє швидко створювати системи медичної інформаційної інфраструктури. Крім того, вона може стати основою або зразком для інших систем. Також дана система має перевагу в тому в порівнянні з іншими системами, що вона має статистичні функції якими може користуватися як лікар так пацієнт.

МЕДИЧНА ІНФОРМАЦІЙНА ІНФРАСТРУКТУРА, ІНТЕГРОВАНЕ
СЕРЕДОВИЩЕ РОЗРОБКИ

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ.....	7
ВСТУП	8
1 МЕДИЧНА ІНФОРМАЦІЙНА ІНФРАСТРУКТУРА АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ.....	10
1.1 Поняття медичної інформаційної інфраструктури.....	10
1.2 Перспективи та можливості розробки подібних систем в Україні.....	11
1.3 Компоненти необхідні для побудови державної медичної інформаційної системи.....	12
1.3.1 Вимоги до бази даних які не залежать від типу даних в цій базі	13
1.3.2 Вимоги до бази даних які залежать від типу даних в цій базі	14
1.3.3 Концепції систем медичної інформаційної інфраструктури.....	15
1.4 Аналіз медичної інформаційною інфраструктури	16
1.5 Порівняльний аналіз медичних платформ	19
2 КЛАСИФІКАЦІЯ РІЗНОВИДИ ТА ФУНКЦІЇ МЕДИЧНОЇ ІНФОРМАЦІЙНОЇ ІНФРАСТРУКТУРИ МЕТОДИ ЇЇ ПЛАНУВАННЯ.....	22
2.1 Структура роботи загальнодержавної медичної системи.....	29
2.2 Методи планування процесу розробки медичної інформаційної інфраструктури.....	31
3 СЕРВЕРНА АРХІТЕКТУРА МЕДИЧНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ	39
3.1 Дволанкова архітектура.....	40
3.2 Триланкова модель	44
3.3 Вибір необхідної моделі клієнт-сервер, щоб конструювати систему	47
4 РОЗДІЛ ПРОЕКТУВАННЯ ТА РОЗРОБКА МЕДИЧНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ	48
4.1 Мова програмування Python	48
4.2 HTML	50
4.3 CSS.....	52
4.4 Інтегроване середовище розробки ruycharm	54
4.5 Архітектура MVT.....	56
4.6 Проектування медичної інформаційної інфраструктури.....	58
4.6.1 Вимоги до системи медичної інформаційної інфраструктури.....	58

4.6.2 Структура бази даних медичної інформаційної інфраструктури	59
4.6.3 Програмне забезпечення функціонування медичної інформаційної інфраструктури.....	65
4.6.4 Робота системи	74
4.6.5 Логіка взаємодії користувача з системою	75
5 ЕКОНОМІЧНА ЧАСТИНА	81
5.1 Опис ідеї медичної інформаційної інфраструктури	81
5.2 Розробка ринкової стратегії	84
5.3 Склад, чисельність та фонд заробітної плати виробничих працівників	87
5.4 Розрахунок матеріальних витрат.....	89
5.5 Розрахунок вартості спожитих послуг.....	90
5.6 Розрахунок амортизації	92
5.7 Кошторис витрат	93
5.8 Карта стейкхолдерів	94
5.9 Висновки	95
6 ОХОРОНА ПРАЦІ І БЕЗПЕКА У НАДЗВИЧАЙНИХ СИТУАЦІЯХ	96
6.1 Аналіз потенційних небезпек	96
6.2 Заходи по забезпечення техніки безпеки.....	98
6.3 Розрахунок захисного заземлення.....	100
6.4 Заходи безпеки у надзвичайних ситуаціях.....	106
6.4.1 Заходи з пожежної безпеки.....	106
6.4.2 Заходи з цивільного захисту	108
ВИСНОВКИ.....	112
ПЕРЕЛІК ПОСИЛАНЬ	114
ДОДАТОК А.....	117

ПЕРЕЛІК СКОРОЧЕНЬ

MIC	– медична інформаційна система
IDE	– інтегроване середовище розробки
HTML	– мова гіпертекстової розмітки
MTV	– система модель, шаблон, уявлення
СУБД	– системи управління базами даних

ВСТУП

Важливою технологією у сучасному світі є системи медичної інформаційної інфраструктури. Розвиток даних систем обумовлений в першу чергу тим, що сучасний стан промисловості дозволяє в великій кількості виробляти комп'ютерну техніку, що в свою чергу залучає велику кількість людей до користування нею.

На даному етапі розвитку медицини в Україні починають створювати великі державні медичні інформаційні системи, які будуть здатні залучати велику кількість державних установ.

Варто також зазначити, що державна медична інформаційна структура розробляється з можливістю підключення приватних установ, що значно розширює функціонал її використання.

Проте в Україні існують і приватні системи медичної інформаційної інфраструктури, вони в першу чергу призначені для забезпечення функціонування приватних установ, проте також не виключають можливість під'єднуватися до державних медичних систем.

На сьогодні є деяка кількість розроблених систем медичної інформаційної інфраструктури, проте всі ці системи мають, досить складну будову, що зумовлює завантаженість системи та призводить до частих системних збоїв.

Також дані системи часто не функціонують автономно, а розробляються як великі інтеграційні проекти. Що може також бути як недолік.

Медична інформаційна інфраструктура яка розробляється повинна відповідати таким вимогам:

- проста і зрозуміла будова, що підвищує надійність системи;
- чітке розподілення функціональних модулів в системі;
- можливість швидкої модернізації.

Розроблення система відповідає всім цим вимогам вона, вона зручна, зрозуміла, надійна.

Також дана система має високу ступінь модульності тобто вона легко може функціонувати як в автономному режимі та к і інтегрувати в себе інші модулі.

1 МЕДИЧНА ІНФОРМАЦІЙНА ІНФРАСТРУКТУРА АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Поняття медичної інформаційної інфраструктури

Медична інформаційна інфраструктура – це спеціальне програмне забезпечення різних типів відкритості. Дане програмне забезпечення автоматизує процес логістичного, економічного та соціального забезпечення медичної системи. А саме надає можливості автоматизувати процеси видачі, призначення, контролю лікарських засобів. Моніторинг діяльності лікарів. Автоматичне регулювання розкладів. Надання широкого спектру статистичних даних для раціонального планування роботи медичного закладу.

Іншою дуже важливою функцією системи медичної інформаційної інфраструктури є дуже велика інформаційно-з'єднувальна функція, яку дана система може надати закладам охорони здоров'я розташованим в різних регіонах, областях, містах. Варто зазначити, що системи даного типу при широкому впровадженні здатні дуже чітко врегулювати роботи медичного закладу та запису людей до нього.

Особливу увагу треба звернути на можливість ведення електронної картки пацієнта. При цьому треба враховувати юридичні аспекти управління персональними медичними даними людини. Спланувати систему захисту персональних даних людини, виявити потенційно нестійкі місця системи, та вжити необхідних заходів.

З попереднього пункту випливає, ще одна дуже важлива функція електронної медичної картки людини, а саме оперативна можливість інформувати лікаря стосовно даних конкретної людини, бо на відміну від паперових карток доступ до електронної здійснюється дистанційно. Уніфікація системи потребує виробити певні стандарти, які зможуть дати

ефективні інструменти розробки та управління медичною інформаційною інфраструктурою.

Які в свою чергу зможуть у такий термін виправляти помилки та системні протиріччя, який дозволить змінювати систему не порушуючи поточної роботи медичної інформаційної інфраструктури.

Адже треба врахувати, що медична інформаційна інфраструктура може буде дуже високо навантаженою системою, яка працює з людьми і основоположним завданням є забезпечення устійливої роботи даної системи.

Необхідно розробити методики інтеграції медичної інформаційної інфраструктура з іншими системами. Знайти такі точки інтеграцій при якій система залишається стабільною. Причому різні інтеграційні компоненти задіяні в медичній інформаційній інфраструктурі мають бути такими, що не суперечать, не розбалансовують, систему в цілому та інші інтеграційні компоненти.

1.2 Перспективи та можливості розробки подібних систем в Україні

Сучасне суспільство дуже тісно пов'язане з інформаційними технологіями, технології подібного типу потребують спеціальної розробки та впровадження. Для того щоб розробити медичну інформаційну інфраструктуру потрібно мати відповідну науково технічну, та матеріально-технічну базу. При розгляданні України за час її незалежності в Україні дуже стрімко розвиваються інформаційні технології. Отже є всі підстави для розробки таких систем.

Варто врахувати взагалі зростаючу роль інформаційних технологій в суспільстві. З розвитком даних систем з'явилася можливість покращити результат обслуговування у медицині за допомогою застосування комп'ютерних технологій, що забезпечує підвищення якості логістичної роботи медичної установи.

Інформаційні системи, призначені з метою вирішення широкого кола завдань у рамках охорони здоров'я, поєднують у собі багато підсистем.

В нашій державі є велика кількість управлінських установ які мають займатися розробкою та впровадженням систем, які призначенні покращити медичну галузь України.

Для цього необхідно проводити широкі консультації з лікарями та медичними установами які зацікавлені в створенні даних систем.

Треба досить чітко розробляти технічне завдання для кожного вузла системи, аналізувати його поведінку в зв'язці з іншими системами. необхідно створювати консультаційні центри які повинні включати в себе технічних фахівців та менеджерів розробки, які повинні дивитися на досвід впровадження подібних систем та аналізувати їхні компоненти їх доцільність в наших умовах.

1.3 Компоненти необхідні для побудови державної медичної інформаційної системи

Необхідною компонентною створення державної медичної інформаційної системи є насичення медичним закладів комп'ютерною технікою. Причому до комп'ютерної техніки повинен висуватися ряд вимог зокрема:

- комп'ютерна техніка повинна бути надійною у використанні мати запас міцності так ,як працювати вона буде практично постійно;
- деталі даної техніки повинні бути легко взаємозамінними.

Також варто налагодити договори з постачальниками такої техніки, техніка обов'язково має пройти гарантійний огляд та сертифікацію. Варто розробити логістичні маршрути та сервісні центри за допомогою яких вчасно можна буде обслуговувати дану техніку, що в свою чергу дозволить стабільно працювати системі.

Необхідно визначити фінансові потреби для забезпечення медичних установ комп'ютерною технікою. Враховуючі великі обсяги її закупівлі.

Одним із основних компонентів медичної інформаційної системи повинні бути бази даних лікарів. У подібних базах даних зберігається інформація, отримана з різних джерел у різний час. Проте за допомогою порівняння, доповнення та фільтрації даних, можна отримати необхідні дані. Також обробляючи інформацію, отриману з відкритих баз даних, засобів масової інформації, а також використання прорахунку осіб, які визначають ступінь важливості та конфіденційності окремих явищ, процесів, фактів можна отримати інформацію. Також можливе використання SQL Injection це технологія злому, яка додає в параметр динамічного SQL не потрібне значення, а деякий код SQL. Ця технологія особливо небезпечна тим, що будь-хто хто має доступ до БД або Web-сайту організації, що звертається до БД, та здатний вводити дані у текстові поля, потенційно може стати джерелом атак за допомогою ін'єкцій SQL. На підставі цього треба виробити певні вимоги до бд.

До прикладу можна проаналізувати вразливості і на підставі цього можна виділити залежні та незалежні від даних заходи забезпечення безпеки баз даних.

1.3.1 Вимоги до бази даних які не залежать від типу даних в цій базі

Функціонування у довіреному середовищі. Довірене середовище це інфраструктура в даному випадку клініки та її захисні механізми, зумовлені правилами безпеки даної медичної інформаційної інфраструктури. Тобто це, функціонування СУБД відповідно до правил безпеки, що застосовуються і до всіх інших систем клініки.

Організація фізичної безпеки файлів даних. Вимоги до фізичної безпеки файлів даних СУБД загалом стандартні.

Вони не відрізняються від вимог, які застосовуються до будь-яких інших файлів даної медичної інформаційної інфраструктури.

Організація безпечного та актуального налаштування СУБД. Ця вимога включає загальні завдання безпеки, такі як встановлення актуальних оновлень, відключення невикористовуваних функцій або застосування ефективної політики паролів, розмежування прав доступу.

1.3.2 Вимоги до бази даних які залежать від типу даних в цій базі

Безпека користувача ПЗ. Сюди можна віднести завдання побудови безпечних інтерфейсів та механізмів доступу до даних.

Безпечна організація та робота з даними. Питання організації даних та управління ними є ключовим у системах зберігання інформації. У цю сферу входять завдання організації даних із контролем цілісності та інші, питання для забезпечення функціонування СУБД. Фактично це завдання включає в себе основний обсяг залежних від даних уразливостей та захисту від них.

Наступною дуже важливою компонентною для державної медичної інформаційної інфраструктури є створення системи адміністрування, контролю та планування діяльності конкретною медичною установою, що повинна в себе включати можливості:

- забезпечення вирішення задач, розвиток яких важко прогнозувати;
- моделювання та аналізу;
- дозволяють легко змінювати постановки адміністративного запиту і вхідні дані;
- відрізняються гнучкістю та легко адаптуються до зміни умов ;
- повинні бути орієнтовані на користувача.

1.3.3 Концепції систем медичної інформаційної інфраструктури

Тут варто розробити певну концепцію при якій ми маємо два варіанти. Перший передбачає створення регіональних систем адміністрування причому вони не повинні співвідноситися з центральною системою в прийнятті рішень і відповідно діяти автономна як в програмному так і апаратному плані.

Другий варіант в свою чергу передбачає повністю централізоване управління як в програмному так і в апаратному плані, причому в даному випадку буде повна регіональна узгодженість.

Якщо обираємо перший варіант відповідно матимемо змогу зробити систему більш стійкою як конкретну регіональну так і в цілому загальнодержавну. В другому випадку система буде більш керованою з центру проте треба врахувати більшу кількість системних збоїв.

Ще однією компонентною системи повинна бути підсистема для конкретної установи яка могла б діяти цілком автономно з наступними завданнями:

- створення індивідуальних графіків роботи лікарів;
- аналізом бази пацієнтів;
- можливість робити електронні картки.

Варто зазначити що всі системи повинні відповідати критеріям захищеності і це стосується не лише баз даних. Потрібно виробити певні протоколи обміну даних суто для таких систем щоб максимально їх убезпечити.

Для цього треба структурувати всі вимоги кожного конкретного закладу, визначити плани по яких відбуватиметься розробка систем захисту.

Окремо треба передбачити можливості для інтеграції цих систем з іншими подібними системами а саме:

- базами даних пацієнтів;
- базами даних лікарів;

- базами даних юридичних осіб;
- системами адміністрування складами;
- логістичними системами.

1.4 Аналіз медичної інформаційною інфраструктури

Проаналізувавши розвиток медичної інформаційної інфраструктури в Україні можна побачити наступне: системи даного типу почали активно розробляти та впроваджувати комерційні організації. Як правило це були великі аптечні мережі. Найпершими потрібні були системи обліку медичних препаратів на складах адже це полегшувало логістичну доставку. По друге в зв'язку зі зростаючою номенклатурою аптечного товару він же препарат ускладнилася система обліку в середині структурного підрозділу. Це завдання почалося вирішуватися зі впровадження в структурні підрозділи звичайних комп'ютерів. Для яких почали впроваджувати системи внутрішньо-структурного обліку товарів та зовнішнього обліку товарів. Це дуже впорядкувало роботу аптечних мереж.

Варто зосередитися також на таких деталях як інтеграція такої системи для аптек з іншими системами. Досить часто зустрічається таке, що система структурного підрозділу, не корегується з системою складського обліку препаратів. Вище було зазначено про необхідність балансу багатьох компонентів в одну систему.

При створенні таких систем дуже важливо правильно налаштувати серверну компоненту.

Також дані системи активно використовують підтвердження та перевірку даних за допомогою мобільного оператора. При аналізі даної системи були помічені збої в роботі мобільно-операторської компоненти, що в свою чергу виводить технічні вимоги для розробки на рівень узгодженості з мобільними провайдерами. Попри мобільно-операторські неузгодженості

також це стосується і медично інформаційної інфраструктури де треба працювати над вдосконаленням алгоритмів обробки мобільних номерів

Другим замовником таких систем стали приватні клініки та кабінети. Звісно системи попереднього типу як для аптек та їх складів не могли забезпечити функціонування таких установ. Почалося розроблення та проектування систем для різних клінік та установ.

Характеризувати ці системи можна великим різномаяттям технічних рішень.

Такі системи в базовому компоненті повинні були містити список лікарів і можливість до них записатися. Відповідно установа могла корегувати час та дату прийому пацієнта. Медична інформаційна інфраструктуру для таких установ та організацій не потребувала впровадження високонавантажених систем що було обумовлено тим, що:

- покриття мережі інтернет мало різну відсотковість на території України;
- початковий стан розвитку медичної інформаційної інфраструктури;
- велика частина навіть приватних установ не розглядала за необхідності перехід на системи медичної інформаційної інфраструктури.

Проте як згодом виявилось це стало набирати обертів відповідно системи таких клінік мали реорганізовуватися в більш навантажені.

Особливість початкових систем було те, що вони не потребували електронної картки людини. Незважаючи на те що кількість людей які користувалися даною системою збільшувалася. Як правило приватні клініки залишали карту паперовою. Також ці системи лише згодом почали взаємодіяти з мобільно-операційним компонентом.

Одна установа мала власну медично інформаційну інфраструктуру. Взаємодії з іншими системами не було.

Захист персональних даних в таких системах проводився на початковому рівні що необхідно було змінювати.

Згодом стало зрозуміло що треба робити системи більшого масштабу, це було пов'язано з такими факторами:

- приватні мережі клінік та установ почали масштабувати свою мережу;
- кількість людей які почали користуватися можливістю дистанційного запису почала збільшуватися;
- загальна інформатизація країни.

Основним рушієм розроблення масштабних систем стали приватні установи. Поширеною практикою стало збільшувати функціонал власної медично інформаційної інфраструктури.

Треба виділити такий аспект, що паралельно з великою кількістю комерційних клінік почало з'являтися багато приватно практикуючих лікарів. Дана категорія теж почала активно включатися в процес створення медично інформаційної інфраструктури. Відповідно логічним кроком було створення систем які могли об'єднувати різних лікарів незалежно від їх спеціалізації.

Платформи даного типу стали основою для розроблення масштабної медично інформаційної інфраструктури.

Наразі в Україні функціонує декілька великих платформ. Це такі платформи як helsi, likarni, doc.ua.

Загалом ці платформи можна характеризувати наступними чинниками:

- велика кількість зареєстрованих в системі лікарів;
- велика кількість відвідувачів;
- співпраця з державними та приватними закладами;
- можливість надання електронного рецепту;
- ведення електронної картки пацієнта.

1.5 Порівняльний аналіз медичних платформ

Аналіз проводить по найбільшим медичним системам які наразі діють та можуть співпрацювати з великою кількістю користувачів.

Проведений аналіз відобразив функції які притаманні певній медично інформаційній інфраструктурі дає змогу поглянути на найбільш характерні риси кожної системи та зрозуміти в якому напрямку треба розвиватися далі. Характеристика наведено таблиця 1.1.

Таблиця 1.1 – Порівняльна характеристика медичних платформ

Назва функції	Назва медичної інформаційної інфраструктури		
	Helsi	Health24	likarni
Можливість реєстрації по номеру телефона	+	+	+
Можливість реєстрації за допомогою інших сервісів	—	—	+
Можливість оцінити лікаря	+	+	+
Співпраця з клініками	+	+	+
Співпраця з приватними лікарями	+	+	+
Співпраця з клініками державної власності	+	—	—

Продовження таблиці 1.1

Назва функції	Назва медичної інформаційної інфраструктури		
	Helsi	Health24	likarni
Можливість купити ліки	+	—	—
Можливість укласти страхування	+	—	—
Можливість декларації з лікарем	+	—	—

Провівши порівняльний аналіз систем медичного призначення, які наразі діють в Україні, можна побачити що в них в порівнянні з попередніми етапами змінилася концепція. Тепер системи даного типу намагаються охопити не конкретну клінічну мережу, а діяти на території всієї України.

Причому варто зазначити, що темпи приросту нових відвідувачів в даних сервісів досить великі. Це пов'язано з тим що деякі платформи почали використовуватися на офіційному рівні відповідно всі державні медичні заклади почали користуватися їхніми послугами.

Також в Україні функціонує медична інформаційна система Doctor Eleks – дана система дозволяє зробити роботу в поліклініках більш автоматизованою, причому характерною особливістю цієї системи є можливість вільного масштабування. В основному система працює з приватними медичними установами, великими мережами. Розробник – компанія Eleks (Львів, Україна). Система Doctor Eleks включає наступні можливості:

- централізоване зберігання інформації;
- швидка та зручна система планування візитів;
- оперативне реагування на телефонні дзвінки;

- контроль за обслуговуванням пацієнтів;
- швидке формування звітів для керівництва.

Також в системі функціонує електронна медична картка пацієнта, механізми редагування шаблонів системи, можливість встановлення графіків візитів, окремо варто зазначити, що система передбачає роботу з фінансовою інформацією та формування фінансових звітів.

Отже як показав аналіз медичної інформаційної інфраструктури в Україні, що як правило такі системи мають чітку компонентну розмежованість. Кожному компоненту відповідає свій функціонал.

Відповідно проектувати дипломну систему будемо теж за принципами чіткого розділення функціональних обов'язків на модулі.

Для цього проведемо планування системи та визначимо основні функції які вона повинна буде виконувати та кожна її частина.

2 КЛАСИФІКАЦІЯ РІЗНОВИДИ ТА ФУНКЦІЇ МЕДИЧНОЇ ІНФОРМАЦІЙНОЇ ІНФРАСТРУКТУРИ МЕТОДИ ЇЇ ПЛАНУВАННЯ

Загалом медичні системи електронного типу можна поділити на декілька типів причому класифікувати можна за різними ознаками наприклад за підпорядкованістю:

- загально державні;
- регіональні;
- міські;
- районні.

Завданнями системи загально державного рівня повинні бути наступні:

- координація та керування всіма службами, та ланками які задіяні в державній системі охорони здоров'я. Планування, роботи окремих регіональних підрозділів;

- співпраця з постачальниками медичного обладнання, визначення необхідних запасів даного обладнання;

- збір статистичних даних з регіональних центрів;
- координація з іншими державними службами.

Завданнями системи регіонального рівня повинні бути наступні:

- планування роботи медичних установ сконцентрованих у регіоні;
- отримання статистики з міських медичних установ;
- координація міських медичних служб;
- ресурсне та медичне забезпечення регіональних медичних установ;

- планування персоналу регіональних медичних установ;
- фінансове планування медичних служб;
- координація навчальних закладів;
- розробка та впровадження перспективних планів роботи;
- взаємодія системи з загально державною;

- співпраця з іншими регіональними центрами.
- планування роботи міських медичних установ;
- матеріально технічне та медичне забезпечення міських медичних установ;
- фінансове планування міських медичних служб;
- координація міських амбулаторій;
- отримання статистичних по ліках, відвідувачах, фінансах;
- співпраця з міськими навчальними закладами медичними;

На рисунку 2.1 зображена ланка системи центр-регіон.

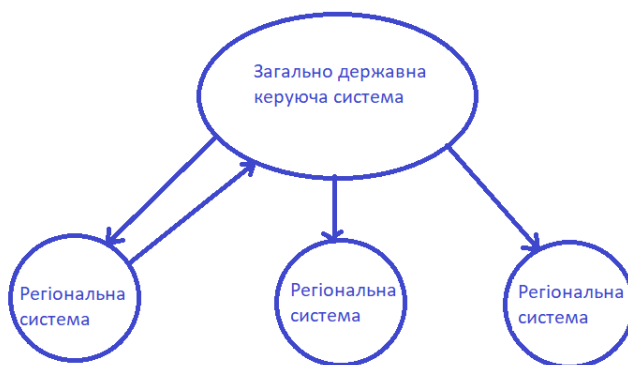


Рисунок 2.1 – Ланка системи центр-регіон

На рисунку 2.2 зображена ланка системи регіон-місто.



Рисунок 2.2 – Ланка системи регіон-місто

Медичні інформаційні системи первинної медичної ланки. Системи даного типу призначені для автоматизації роботи медичних служб первинної ланки. Це окремі лікарські кабінети, лабораторії, реєстраційні пункти центрів первинної медико санітарної допомоги, системи даного типу можуть бути спеціалізовані під лікарські потреби певної галузі. Основними завданнями систем даного типу є:

- надання широко спектру інформації терапевтам загальної практики та сімейним лікарям;
- виписка та надання електронного рецепту відвідувачу, в значно швидші терміни ніж в паперовому вигляді;
- ведення електронного реєстру;
- надання лікарю базових статистичних даних.

Окремо виділяють медичні інформаційно пошукові – системи. Системи даного типу призначені для пошуку необхідної інформації, чи документу, або рішення. Вони не мають на меті зміну інформації. Як правило вони виконують роль модуля в більш ширшій системі, і в середній медичній установі ними може користуватися звичайний лікар. Їх завдання це:

- пошук медичної інформації;
- пошук статистичних файлів;
- пошук нормативних документів даної установи.

Інформаційні системи галузевого пошуку. Наразі медична наука досягла досить великої спеціалізації відповідно, медична інформація, наукові медичні статті та праці, дослідження в галузі медицини набули більш конкретизовано, та галузевого значення. Постає завдання розробки систем диференційованого пошуку, що значно покращить інформаційно-медичну комунікацію. Основними завданнями таких систем є:

- пошук спеціалізованої медичної інформації;
- пошук галузевих медичних досліджень;
- пошук конкретизованих медичних статистичних даних;
- розробка відповідних тематичних довідників.

Лікарські інформаційно-діагностичні системи. Як правило системами даного типу призначені для автоматичного визначення різних медичних станів, данні системи мають широку класифікацію ознак за якими можна визначити стани.

Такі систем можуть визначати діагноз, причому визначення діагнозу може також супроводжуватися рекомендаційними висновками що до лікування.

Методами визначення діагнозу в таких системах служить ймовірнісний підхід який ґрунтується на статистичних даних. Варто зазначити що діагнози таких систем не можна вважати остаточними, адже система не може врахувати великої кількості симптомів. Тому в таких системах потрібен лікарський висновок що до, діагнозу.

Такі системи ще називають експертними. У них є ряд переваг які допомагають працювати лікарю це зокрема:

- швидкість визначення діагнозу;
- зв'язка симптоматики з іншими подібними діагнозами;
- можливість більш точного визначення діагнозу.

Медичні системи управління та планування. Системи даного типу призначені для планування роботи окремої медичної установи, чи медичних систем в конкретному регіоні, або районі. Такі системи повинні планувати:

- постачання лікарських засобів в медичні заклади;
- постачання матеріалів необхідних в господарстві;
- постачання медичних комплектуючих;
- фінансове забезпечення медичної установи;
- статистичне забезпечення;
- взаємодія та корегування планів у відповідності з іншими системами;
- розробку оптимальних планів роботи на основі статистичного аналізу;
- розробка альтернативних планів дій.

Персональна комп'ютерно-інтегрована система лікаря. Вона застосовується для завдань поєднання різної діагностичної техніки в єдиний комплекс. Дозволяє лікарю-спеціалісту автоматизувати робочий процес отримання, обробки, та інтерпретації інформації медичного призначення.

Такі системи широко застосовуються в медичних діагностичних центрах. Вимагають спеціального програмного забезпечення. Повинні з'єднуватися з системами баз даних регіонального, районного, та державного рівнів. Причому бази даних повинні бути спеціалізованими.

Персональна комп'ютерно-інтегрована система лікаря повинна відповідати таким вимогам:

- робити диференційовані пошуки медичної інформації;
- алгоритми виявлення типових рішень для лікаря;
- можливість комунікувати з іншими установами;
- створення статистичних даних галузевого призначення;
- можливість інтеграції з іншими системами;
- мати систему захисту.

Наприклад Персональна комп'ютерно-інтегрована система лікаря кардіолога. Данна система повинна бути інтегрована з кардіографом. Також підключена до бази даних інших кардіографічних центрів, з метою проведення статистичного аналізу, та накопичення даних для корекції роботи прогнозуючих систем лікаря.

Данна система складатиметься з:

- кардіографа;
- комп'ютера;
- системи підключення до бази даних;
- аналітичної системи.

Варто додати, що кожна персональна комп'ютерно-інтегрована система лікаря повинна бути інтегрована в загальну систему медичного закладу, для отримання та надання більш чіткої інформації що покращить роботу медичного закладу.

Медико-реєстраційні інформаційні системи. Головним завдання таких систем є забезпечення запису, прийому, відвідувачів медичного закладу, а також координація при створенні робочого графіку, керівників відділень, структурних медичних підрозділів. Данні системи також мають сприяти налагодженому, та упорядкованому функціонуванню медичних закладів.

Вони складаються з таких частин:

- база даних з графіками структурних підрозділів;
- база даних відвідувачів.

Їх основні функції:

- мають забезпечувати реєстрацію нових відвідувачів;
- при необхідності узгоджувати роботу з іншими медичними установами;
- робити статистичне прогнозування.

Системи контролю медичного та хоз майна. Як показала практика неузгодженість у плануванні призводить до того, що одні підрозділи отримують надлишкові медичні ресурси, а інші навпаки можуть не отримати, або отримати те чого не замовляли. Відповідно на такі системи покладаються завдання:

- аналіз поточного стану медичних ресурсів;
- збирання запитів на медичні ресурси з інших підрозділів;
- координація з ресурсними базами інших медичних підрозділів;
- координація з центральними складами постачання;
- раціональне планування використання медичних ресурсів.

Також як підсистема даної системи, виступає система фінансового моніторингу секторів (певних напрямків товарів на ринку), яка дозволить більш раціонально закуповувати необхідні медичні ресурси, що дозволить оптимізувати використання медичного бюджету конкретної установи. або міста.

Данна система повинна працювати і інтеграції з системами фінансового планування, для покращення загальної роботи рисунок 2.3.

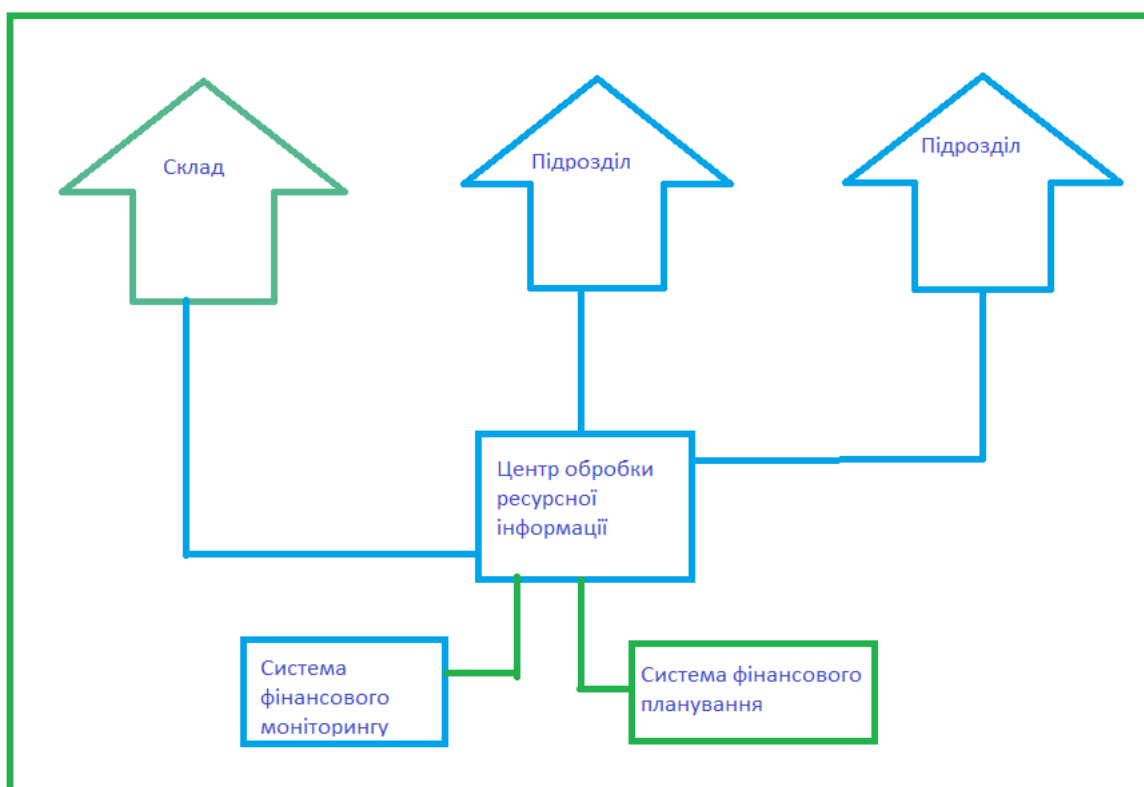


Рисунок 2.3 – Система контролю медичного, та хоз майна

Варто зазначити, що необхідним компонентом медичної інформаційної інфраструктури є системи кадрового планування. Які повинні займатися аналізом структури зайнятості медичних працівників. Визначати необхідні потреби в медичних працівниках. Данні системи також повинні взаємодіяти з загальнодержавними медичними системами. Повинні виявляти потреби медичних працівників аналізувати їх поточний та перспективний стан. Такі системи повинні в себе включати:

- бази даних медичних працівників;
- статистичні данні з медичних закладів про кількість працюючих, частоту зміни персоналу, та інші данні для формування необхідних планів;
- системи координації з медичними закладами всіх регіонів, для розроблення планів оптимізації розподілення медичних кадрів;
- аналізувати можливостей для покращення роботи працівників.

2.1 Структура роботи загальнодержавної медичної системи

Етап перший.

На першому етапі загальнодержавна система визначає ряд медичних потреб зокрема: планує кількість необхідних в державі лікарень, визначає кількість профільних медичних закладів. За необхідності проводить географічне районування. Далі за допомогою інших державних служб та використання статистичних даних, система планує логістичне забезпечення.

Логістичне забезпечення варто розпочати з визначення регіонів та їх географічного розташування.

Наступним чином проводять логістичне планування району, та всіх населених пунктів в середині району.

Після проведення логістичного планування, проводиться інфраструктурний аудит де система визначає, стан інфраструктури, необхідність її модернізації, або необхідність побудови нової.

Другий етап.

На цьому етапі починає працювати системи контролю медичного та хоз майна. Завдання цього етапу є проведення аудиту та формування чіткого плану постачання конкретної медичної установи, виведення статистичних даних використаних медичних ресурсів.

Далі система проводить фінансовий аудит, визначає необхідний бюджет для фінансування конкретної медичної установи, спеціалізованої медичної установи, або медичної галузі.

Наступним кроком буде визначення ресурсно-матеріальних потреб, конкретного району, регіону і в цілому держави.

Третій етап.

Проведення кадрового аудиту медичної установи. Кожна медична установа, в районі чи регіоні, проводить необхідне планування по визначенню необхідної кількості працівників використовуючи системи кадрового планування.

Після чого проводить загальне планування працівників.

Четвертий етап.

Далі на рівні кожної установи починають діяти медичні інформаційні системи первинної медичної ланки. Вони повинні надати можливість автоматизувати велику кількість рутинних процесів.

Далі підключаються інформаційно-пошукові системи, причому вони повинні носити інтеграційний тип.

Також необхідно підключити інформаційні системи галузевого пошуку, бажано щоб вони були підключені до загально державної бази даних, медичної інформації.

В загальну систему лікарні треба інтегрувати лікарські інформаційно-діагностичні системи.

Персональна комп'ютерно-інтегрована система лікаря скомпонує всі інші системи, і повністю сформує цикл медичної інформаційної інфраструктури.

Після встановлення всіх систем ми матимемо медичну інформаційну інфраструктуру у конкретній медичній станіві яка зможе:

- вести облік відвідувачів;
- автоматизувати рутинні процеси;
- формувати план фінансових витрат;
- формувати потреби в медичних ресурсах;
- пришвидшити пошук інформації медичного характеру;
- зробить диференційні системи пошуку;
- пришвидшити обмін інформацією між закладами.

Тепер установа яка володіє. Такою медичною інформаційною інфраструктурою буде більш ефективно працювати, планувати роботу окремих структурних підрозділів, вести облік відвідувачів. Матиме змогу поліпшувати медичні знання працівників за рахунок швидкого обміну медичною інформацією.

2.2 Методи планування процесу розробки медичної інформаційної інфраструктури

Канбан – це метод підвищення ефективності управління. Це принцип, система організації роботи в команді розробників медичної інформаційної інфраструктури або іншого продукту. Він заснований на гнучкості управління процесом та направлений на результат. Саме результат – якісний і точно вчасний – мета будь-якої команди, яка працює по канбан (недаремно його вигадали японці).

Мета канбан – зробити проект зрозумілим, наочним, відстежити готовність робіт та проконтролювати графік спеціалістів які його розробляють спеціалістів.

Для покращення розуміння та контролю робочий процес візуалізують на дошці, поділеній на колонки. Кожна колонка – це поточний стан робіт. Безпосередньо кожне завдання відображають у канбан-картках – там можна прочитати їх опис, рівень пріоритетності та додаткову інформацію. Коли завдання завершує певний етап, картку з описом даного завдання переносяться у наступну відповідну колонку. Поглянувши на дошку, можна відразу зрозуміти, як ситуація з проектом. Дошка канбан зображена на рисунку 2.3

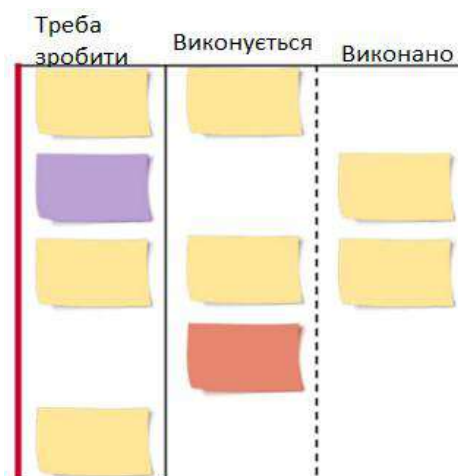


Рисунок 2.3 – Дошка канбан

Канбан-дошки можна будувати за різними принципами. У першому випадку це звичайна дошка зі стовпцями. Завдання пишуть на стікерах і наклеюють у потрібний розділ, по мірі виконання завдання переміщуючи. Електронні дошки мають аналогічні функції, але в порівнянні з фізичним проте можуть мати значно більше функцій моніторингу, та бути більш гнучкими у використанні.

Загалом в базовому варіанті дошки кан-бан мають три стовпці:

- треба виконати;
- виконується;
- виконано.

Також стовпці у дошки можна конкретно обрати під кожний проект.

Система канбан базується на 9 принципах: прозорість, баланс, співпраця, клієнтоорієнтованість, потік, лідерство, розуміння, згода, повага.

Прозорість показує поточний стан справ у межах роботи над конкретною задачею. Дозволяє контролювати метрики і на підставі них приймати рішення щодо, подальшої діяльності. Використовуються лише чітка прості та зрозумілі всім термінологія.

Баланс між очікуваннями які закладають в розробку та її реальними можливостями. Не дозволяє завищувати очікування над можливостями, щоб не допустити вигоряння працівників та велику перепрацювання. Так само не применшити очікування, щоб робота над завданням не простоювала. Цей дозволяє більш ефективно використовувати ресурси компанії Варто враховувати, що співпраця яка дозволить знайти взаємодію, для покращення роботи команди.

В даній системі потік це послідовність дій, які дозволяють перетворити задум на реальну задачу.

Лідерство здатність вести за собою оточуючих тобто канбан команду. У канбан команди повинен бути її лідер. Розуміння і його цінність канбан-метод бачить у тому, щоб зрозуміти існуючі ролі, існуючі процеси та причини їх появи. І орієнтуватися на них.

При розробці проекту з використанням канбан треба спочатку визначитися з загальною кількістю задач. Кожна задача має вписуватися в конкретний блок ,підсистему і загальний проект, при необхідності дозволено доповнювати задачі, або їх змінювати. Задачі конфігуруються відповідно до пріоритету пріоритету. Рівень пріоритетності визначає сама команда. Пріоритети можна змінювати по ході виконання задачі – це одна з особливостей канбану.

Другим кроком буде розбиття задач між розробниками. Небажано нерівномірно та нераціонально розподіляти задачі між розробниками. Допустимо виставлено обмеження з одного завдання на людину. І в разі виконання розробник залишається незадіяним. Якщо, число завдань, що виконуються одночасно одним розробником, необмежене – розробник починає виконувати одне завдання, потім друге, третє. У результаті багато задач які розробляються але нема виконаних. Виходить, що низький ліміт незавершених завдань погано відбивається на продуктивності, а високий – дає змогу уповільнити проект. Необхідно підібрати обмеження, баланс який дасть рівномірне завантаження.

Канбан забезпечує свободу прийняття рішень. Якщо при встановленому обмеженні не вдається вкластися у графік, змініть графік. Коли розробники мають багато вільного часу – ліміт можна збільшити.

Перевагою канбан є то, що він дозволяє варіативність планування. Розробники сконцентровані на поточних процесах, але при зміні завдання або внесені коректив можна швидко змінити розподіл задач.

Взаємодія в середині команди. Спільне обговорення всіх питань та пошук оптимальних рішень покращують колективу. Кожен співробітник розробник розуміє, що від може планувати проект.

Оптимізація роботи. За рахунок того, що можна звернутися за допомогою до колег при вирішенні задач, покращується виконання роботи.

Команда завжди бачить, у кого завдання не йде і може допомогти, щоб тим саме покращуючи процес.

Швидке виявлення проблем. Завдяки розподіленості проблемні місця одразу помітні. Пошук оптимальних рішень можна направити саме на такі місця.

Наочність. Робочі процеси є абсолютно прозорими, оскільки будь-який співробітник легко може переглянути поточні етапи та статуси завдань.

Проте канбан гарно працює тільки коли чисельність команди в розмірі 5-6 розробників. Бо це дозволить зробити комунікацію швидкою і зрозумілою.

Команда розробників повинна працювати над спеціалізованим продуктом, тобто команда працює на один результат і може переключатися на весь спектр завдань, на відміну від великих проектів де є чітке розподілення за можливістю виконувати завдання.

Також канбан приміняють, як правило для конкретного проекту, який не є багатокомпонентним.

Команді варто дотримуватись практичних порад канбан.

Вчасно переміщайте завдання змінюйте статуси завдання та відстежуйте рух.

Обговоріть правила роботи, команда повинна чітко розуміти, як поводитися з дошкою, коли можна брати нові завдання, що робити при виникненні складнощів, як визначати готовність роботи.

Обмежте число завдань, що одночасно виконуються розробник непродуктивно витрачає сили, якщо паралельно веде безлічі задач або, навпаки, простоює через нерівномірне завантаження. Сплануйте із командою, яку кількість робіт з кожного етапу оптимально вести одночасно, проставити пріоритети.

Збирайте розробників та обговорюйте нюанси роботи, успіхи та невдачі. Розробники самі можуть планувати зустрічі. Збиратися щотижня або зустрічатися раз на місяць, або збиратися онлайн за необхідності. Треба вирішити, як оптимізувати процеси. Додатково обговорюють інші нагальні питання.

Також треба дотримуватись пошуку оптимальної системи. Ціль – прискорити рух карток по дошці. Розробники повинні постійно проводити якісь зміни, вдосконалювати пріоритети та інше. Щоб система дійсно змінювалася, зміни потрібно вводити для всієї команди разом. Зміни проводити поетапно.

Scrum – це метод планування проекту. Він гнучкий і може застосовуватися до різних типів проектів.

Для роботи необхідно сформувати групу розробників. Це група людей, які працюють над проектом. Кількість розробників від трьох до дев'яти осіб. Коли розробників стає більш то їх можна об'єднувати у скрам команди.

Згідно із концепцією скрам, команда має працювати коротким циклами, які називаються спринти. Спринти мають фіксовану тривалість і йдуть один за одним без перерв та проміжків. Довжина спринту визначається типом проекту який робиться, та корегується розробниками, і зазвичай становить від одного до чотирьох тижнів.

Після кожного спринту команда повинна представити результат ,який вона змогла виконати за певний спринт.

Product Owner відповідає за фінішний результат. Саме керівник проекту ухвалює остаточні рішення за результатами нарад розробників, комунікує із замовником. Як правило команди обирають одного керівника проекту, це полегшує координацію. Проте один керівник може керувати кількома проектами.

Скрам-майстром керує однією командою розробників команди. Він відповідає за результат роботи проекту, а за весь проект відповідає керівник проекту. Основна функція скрам-майстра це координація всіх членів команди, допомога керівник проекту у розподілі ресурсів, контролі.

Всі ідеї та побажання розробники продукту складають є у Product Backlog. Фактично це документ накопичення ідей.

Беклог продукту – це впорядкований перелік усіх побажань та ідей, над якими працюватиме скрам-команда. Подивившись у беклог продукту, будь-якій зацікавленій особі має стати зрозуміло, що й у якому порядку робитиметься. Керує беклогом власник продукту.

Етапи які будуть виконуватись у спринті, плануються на нараді з планування спринту. Нарада з планування спринту триває щонайбільше чотири години протягом двох тижнів спринтів і вісім годин протягом одного місяця. Керівник проекту несе відповідальність за забезпечення того, щоб збори проходили, забезпечує присутність команди, та пояснює мету запланованої наради. Майстер Scrum модерує зустріч, підтримує дискусію і закриває зустріч.

Команда обговорює функціональні можливості, та необхідні дії під час спринту. Власник продукту надає роз'яснення щодо елементів їх конфігурації, та в цілому концепції. Команда розробляє відповідний план та планує спринт.

Далі необхідно визначити мету кожного спринту. Мета спринту – це завдання, яке дає керівник команді, та роз'яснює етапи в середині спринту. Далі команда вирішує, як вона створюватиме обрану функціональність у робочому продукті під час спринту.

Робота під час спринту оцінюється під час проміжного планування. Після планування можна роботу розбити на блоки. Це дозволяє легко розподілити роботу та відслідковувати етапи її реалізації. Якщо команда розуміє, що темпи або знизились, або збільшились вона може корегувати графік.

Команда може залучати інших розробників для участі у нараді з планування, щоб отримати технічну допомогу.

Щоденні зустрічі – це спосіб координувати зусилля команди, оперативно виявляти проблеми, інструктувати щодо виконаних завдань. Учасники збираються біля орієнтуючись на план, і кожен учасник скрам-команди по черзі відповідає на три запитання:

– що я зробив учора? що я зроблю сьогодні? що мене гальмує та заважає рухатися вперед? Час 15 хвилин, тому треба швидко виявити проблеми. Рішення можна розробляти окремо.

Далі скрам-команда, керівник проекту технічні консультанти збираються для огляду спринта. Команда та керівник проекту демонструють результати спринту, розповідають, як пройшов спринт, відповідають на запитання. Далі треба проаналізувати беклог, за необхідності внести корективи, обговорити подальші дії.

Ретроспектива спринту. Концепція скрам передбачає, що скрам-команда буде постійно розвиватися, удосконалюватися і підвищувати свою ефективність. Тому спеціально роблять наприкінці кожного спринту аналітична ретроспектива спринту. Скрам-команда обговорює спринт: що було добре, які виникали питання, що нового можна зробити в процесі роботи. Тут команда може обговорювати всі аспекти: організацію роботи команди, внутрішній процес розробки, технічні методи, що використовуються. Як висновок треба розробити план покращень.

Загалом два попередніх методи можна назвати гнучкими.

Розглядаючи гнучкі методи (Agile), треба проаналізувати негативні так і позитивні аспекти цієї методології.

Данні методи мають велику адаптивність, тобто планування за такими методами гнучке. В традиційній методології планування треба дотримуватись чіткого порядку, то гнучкі методи легко підлаштовується під нові технічні вимоги.

Продукт який розробили має дуже точну функціональність тому, що на кожному етапі він перевіряється.

Проекти даного типу швидко запускається, легко адаптується на зміни, дозволяє команді розробників та клієнтів підтримувати постійний зв'язок та координацію.

Треба враховувати у цій методології те, що, постійний зворотний зв'язок може значно збільшувати час проекту, уповільнення роботи, що може

суттєво вплинути на розробку. Треба регламентувати зворотній зв'язок і чітко дотримуватися плану.

Треба враховувати необхідності адаптувати під умови проекту технічні вимоги та документацію. Налагодити оперативне інформування команди про зміни технічного завдання або додаткові функції що внесені у документи з функціональними вимогами. Щоб не було технічних розбіжностей.

При даному типі планування може виникнути потреба у частих зустрічах. Тут теж треба продумати систему комунікації між членами команди, аби не було постійних відволікань. Це поліпшить роботу.

Також треба врахувати при плануванні гнучким методом необхідність постійної присутності керівника проекту, плани будуються на один спринт чи під конкретний систему, і це потребує професійних спеціалістів.

Проаналізувавши функції які може виконувати система. та визначивши її основні класифікаційні компоненти, було прийнято рішення, що основними модулями в розроблюваній системі будуть модулі пошуку інформації, реєстраційні модулі до лікаря та модуль реєстрації користувача.

Також при плануванні системи будуть використовуватися методики канбан та скрам для ефективної роботи планування, розробки та впровадження медичної інформаційної системи.

3 СЕРВЕРНА АРХІТЕКТУРА МЕДИЧНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

Медична інформаційна інфраструктура це в першу чергу комп'ютерна система. Загалом її можна охарактеризувати схемою клієнт-сервер.

Архітектура "Файл-сервер". Історично перша архітектура інформаційних систем. Як виконувані модулі, так і дані розміщуються в окремих файлах операційних систем. Доступ до даних здійснюється шляхом вказівки шляху (path) та використання файлових операцій (відкрити, рахувати, записати). Для зберігання даних використовується виділений сервер (окремий комп'ютер), який і є файловим сервером. Модулі, що виконуються, зберігаються або на робочих станціях, або на файловому сервері. В останньому випадку спрощується процедура їхнього адміністрування, але при цьому зростають вимоги до надійності мережі [1].

Технологічна схема " клієнт – сервер " передбачає наявність певної кількості комп'ютерів, об'єднаних у мережу, один із яких виконує спеціальні управляючі функції (є сервером мережі).

У базовій моделі клієнт-сервер усі процеси у розподілених системах діляться на дві групи, що можливо перекриваються. Процеси, що реалізують деякий сервіс, наприклад, сервіс файлової системи чи бази даних, називаються серверами. Процеси, що запитують послуги у серверів шляхом посилки запиту та подальшого очікування відповіді від сервера, називаються клієнтами [2].

Клієнт як правило формує запит до сервера, на якому розташована БД, структурною мовою запитів SQL (Structured Query Language), що є стандартом управління реляційних БД [3].

Загалом архітектуру клієнт – сервер можна поділити на двухланкову та багатоланкову.

3.1 Дволанкова архітектура

У інформаційній мережі (навіть одноранговій), побудованій на сучасних мережевих технологіях, є елементи клієнт-серверної взаємодії, найчастіше на основі дволанкової архітектури. Дволанковою (two-tier, 2-tier) вона називається через необхідність розподілу трьох базових компонентів між двома вузлами (клієнтом та сервером).

Це такі компоненти:

- функції введення та відображення даних (забезпечують взаємодію з користувачем); Також цю групу називають логікою уявлення. Данна група дозволяє користувачеві працювати із програмою. Незалежно від конкретних характеристик логіки уявлення (інтерфейс командного рядка, складні графічні інтерфейси користувача, інтерфейси через посередника) її мета така, щоб максимально забезпечити функції для найбільш ефективного обміну даними між користувачем та інформаційною системою [4];
- прикладні функції, притаманні даної предметної області.

Такі процеси також називають бізнес-логікою. Бізнес-логіка окреслює мету, навіщо конкретно призначено додаток (вказує конкретні функції для певної області). Розподіл такого застосування забезпечує логічну основу для розподілу програми на кількох комп'ютерах.

Функції управління ресурсами (файловою системою, базою даних тощо) Функції такого типу називають логікою доступу до даних або алгоритмами доступу до даних. Алгоритми доступу до даних, або його ще можна визначити як специфічний для конкретної програми інтерфейс механізму постійного зберігання даних це можна порівняти з файловою системою або СУБД. Інтерфейс СУБД робиться за допомогою спеціальних модулів призначення яких обробка даних. Мета даного інтерфейсу, щоб програма мала змогу керувати з'єднаннями з базою даних та запитам до неї Тобто робити переклад специфічних для даного додатка запитів на мову

SQL, отримання результатів та переведення отриманих результатів назад у специфічні для конкретної програми структури даних.

Дволанкова архітектура використовується в клієнт-серверних системах, де серверний компонент працює на клієнтські запити безпосередньо та в повному обсязі, причому використовуючи лише власні серверні можливості. Сам. сервер не викликає сторонніх мережових програм і не звертається до сторонніх ресурсів для виконання будь-якої частини запиту.

На рисунку 3.1 зображена архітектура клієнт сервер.



Рисунок 3.1 – Архітектура клієнт сервер [4]

У моделі віддаленого доступу (Remote Data Access, RDA) база даних зберігається на сервері. На сервері знаходиться ядро СУБД. На клієнті розміщується файлова логіка та бізнес-логіка програми. Клієнт робить запит до сервера із запитом на мові SQL.

Переваги даної моделі:

- перенесення компонента представлення та прикладного компонента на клієнтський комп'ютер покращує роботу та оптимізує роботу

сервера БД, таким чином загальну кількість процесів в операційній системі зводиться до мінімуму;

- сервер БД звільняється від невласливих йому функцій; процесор або процесори сервера повністю завантажуються операціями обробки даних, запитів і транзакцій (це стає можливим, якщо відмовитися від терміналів, що
 - не мають ресурсів, і замінити їх комп'ютерами, що виконують роль клієнтських станцій, які володіють власними локальними обчислювальними ресурсами);
- різко зменшується завантаження мережі, оскільки у ній від клієнтів до серверу передаються не запити на введення-виведення у файловій термінології, а запити на SQL, та його обсяг значно менше. У відповідь на запити клієнт отримує лише дані, відповідні запиту, а не блоки файлів, як у FS-моделі [5].

Структура моделі віддаленого доступу наведено на рисунку 3.2.



Рисунок 3.2 – Структура моделі віддаленого доступу [5]

У DBS-моделі сервера баз даних компонент представлення виконується на комп'ютері-клієнті, проте прикладний компонент являє собою набір процедур, що зберігаються, і працює на сервері -комп'ютері- БД. Там

виконується функція доступу до даних, тобто ядро СУБД. Переваги DBS-моделі великі: це і можливість всеохоплюючого адміністрування прикладних функцій, і зниження кількості запитів (замість SQL-запитів застосовуються спеціальні виклики процедур, що зберігаються), і можливість поділу процедурних функцій між декількома додатками, і економія ресурсів комп'ютера за рахунок використання спеціального протоколу запитів процедури. До недоліків моделі можна віднести обмеженість засобів, що використовуються для написання серверних процедур, які являють собою різноманітні процедурні розширення SQL, що не витримують конкуренції з мовами третього покоління, такими як C або Pascal. Сфера їх використання обмежена конкретною СУБД, у більшості СУБД відсутні можливості налагодження та тестування розроблених процедур, що зберігаються.

З розробкою та використанням лише на рівні серверів баз даних механізму збережених процедур з'явилася потреба у концепції активного сервера БД. Таким чином частина функцій прикладного компонента реалізовані у вигляді серверних процедур, що зберігаються на стороні сервера. Решта прикладної логіки яку треба виконати робиться на клієнтській стороні.

Структура моделі сервера баз даних зображена на рисунку 3.3.

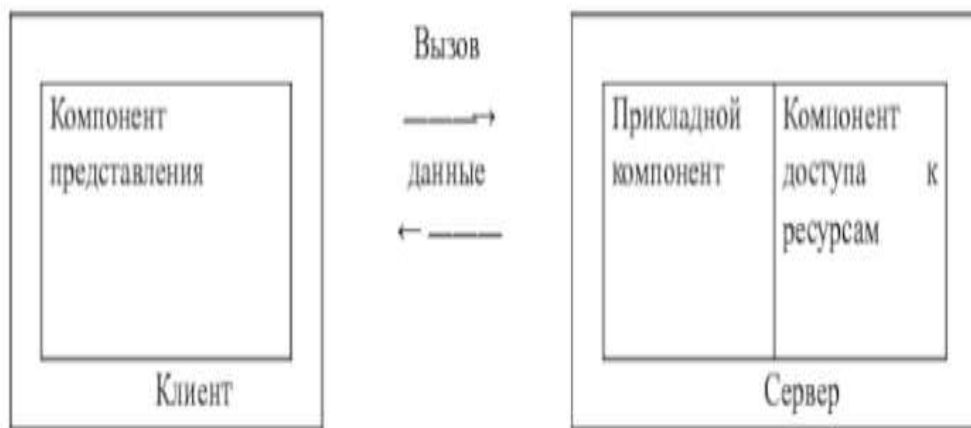


Рисунок 3.3 – Структура моделі сервера баз даних [6]

3.2 Триланкова модель

Також архітектура клієнт-сервер може використовувати технологію триланкової (багатоланкової) моделі яка пов'язана з зростаючим використанням розподілених обчислень. Вони реалізуються на основі моделі сервера додатка, де мережева програма розділена на дві або більше частин, кожна з яких може виконуватися на окремому комп'ютері. Для цього розробляється спеціальні протоколи за якими програми взаємодіють один з одним, обмінюючись повідомленнями. В цьому випадку дволанкова клієнт-серверна архітектура стає триланковою (three-tier, 3-tier) [7].

Багатоланкові архітектури клієнт-сервер є логічним продовженням поділу додатків це можна віднести і до рівня користувача інтерфейсу, також це можуть бути компонент обробки та даних.



Рисунок 3.4 – Триланкова модель клієнт сервер [8]

Різні ланки взаємодіють по чітко визначеній логіці. У багатьох бізнес-додатків розподілена обробка даних фактично еквівалентна організації багатоланкової архітектури додатків типу клієнт-сервер. Розподіл за таким принципом називається вертикальним (ВР). Особливістю вертикального розподілу є те, що він досягається розміщення логічно різних компонентів на різних машинах. Також можливий розподіл на клієнти та сервери який відбувається способом, відомим як горизонтальне розподілення (ГР). Даний розподіл дозволяє клієнту або серверу містити фізично розділені частини логічно однорідного модуля, причому працювати з кожною частиною можна незалежно. Це робиться для вирівнювання завантаження [9].

У даній моделі сервер додатків (Application server, AS) – різновид багаторівневої архітектури "клієнт-сервер", заснований на схемі "клієнт-сервер додатків". Тобто тут вводиться додатковий спеціальний проміжний рівень між частиною клієнта та частиною сервера. Даний проміжний рівень включає в себе один або кілька серверів додатків. Звідси й ідентична назва – модель сервера додатків AS (Application Server).(!).

Тонким клієнтом в серверній архітектурі називають комп'ютер, обчислювальні ресурси якого можуть запустити необхідний мережевий додаток через web-інтерфейс. Інтерфейс користувача такого додатка формується засобами мови HTML.

Щоб він зміг працювати достатньо лише забезпечити можливість запуску web-браузера, у якому і будуть відбуватися всі дії. Також web-браузер через це називають "універсальним клієнтом".

Для того, щоб працювала тонкий клієнт на комп'ютері треба завантажити програму. Коли починається завантаження тонкого клієнта сітвовій карті даного комп'ютера видається окрема IP адреса.

Після закінчення завантаження тонкий клієнт починає термінальну сесію із зазначеним у налаштуваннях термінальним сервером.

При використанні тонкого клієнта можна працювати на недуже потужній комп'ютерній техніці, по суті звичайні комп'ютери можна об'єднати за допомогою термінального сервера, установивши на кожний програму тонкого клієнта. Можна оптимізувати витрати на програмне забезпечення – адже не треба купувати спеціального програмного забезпечення треба буде лише ліцензувати термінальний сервер.

Тонкий клієнт Thick client, це програма, яка виконує всі задані користувачем команди незалежно від централізованого сервера.

Він може бути так званою робочою станцією або персональним комп'ютером. Також вони повинні працювати під управлінням власної дискової операційної системи. З серверами «товсті» клієнти працюють в сновному по схемі клієнт основний, а сервер допоміжний ,наприклад звертаються до корпоративної бази даних.

Також тонкий клієнт може працювати як спеціальний додаток, такий додаток як правило запускається з локальної операційної системи. Він повинен мати компоненту представлення даних та прикладний компонент який зможе реалізувати обчислювальні потужності комп'ютера.

Перевагою даного від попереднього типу є те, що він:

- може працювати автономно незалежно від основного серверу;
- має більший функціонал;
- працює в режимі роботи при якому взаємодіє з багатьма користувачами;

- не залежить від функціональних налаштувань головного серверу тому має більшу надійність;
- висока швидкодія.

Недоліки:

- так як кожен такий комп'ютер фактично автономна машина то його треба окремо обслуговувати;
- кожний такий комп'ютер потребує спеціального апаратного забезпечення, що в свою чергу робить його дорожчим;
- потребує спеціального програмного забезпечення.

Загалом архітектура типу клієнт сервер наразі є наймасовішою в розробці різних проектів.

3.3 Вибір необхідної моделі клієнт-сервер, щоб конструювати систему

Так як наша система буде базуватися на фрейм ворку django який в свою чергу працює за допомогою схемою уявлення, шаблон, модель буде раціонально використовувати триланкову схему клієнт-сервер .

Адже така схема забезпечує використання серверу додатків, який є необхідною компонентною django. Також система може бути розподілена між багатьма серверами, що робить триланкову модель найбільш оптимальною для використання в магістерській роботі.

4 РОЗДІЛ ПРОЕКТУВАННЯ ТА РОЗРОБКА МЕДИЧНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

4.1 Мова програмування Python

Система яка розробляється повинна бути зрозумілою і не лише в плані інтерфейсу, а й в плані читання коду.

Також розроблювана система повинна буди модульною, а мова Python забезпечує таку модульність. Також мова Python використовується при роботі з django який в свою чергу є основою для розробки проекту.

Тому для розробки медичної інформаційної інфраструктури була вибрана мова Python.

Python – це інтерпретована об'єктно-орієнтована мова програмування високого рівня з динамічною семантикою. Його високорівневі вбудовані структури даних у поєднанні з динамічною типізацією та динамічною прив'язкою роблять його дуже привабливим для швидкої розробки додатків, а також для використання як мови сценаріїв або зв'язувальної мови для з'єднання існуючих компонентів [10]. Простий і легкий в освоєнні синтаксис Python забезпечує зручність читання коду і, отже, знижує вартість обслуговування програми. Завдяки тому, що Python підтримує модулі та пакети, програми написані на цій мові набувають модульності, що значно сприяє повторному використанню коду. Інтерпретатор Python та велика стандартна бібліотека доступні у вихідній або двійковій формі для всіх основних платформ і можуть вільно розповсюджуватися.

Серед основних переваг які можна віднести до мови програмування пайтон можна віднести високу продуктивність. Робота з програмами на мові пайтон досить проста, тобто помилка при роботі, або неправильне введення не буде являться фактором сегментації. Варто зазначити, що на даній мові програмування не використовується компілятор як на мові C тому, що він автоматично вбудований в інтерпретатор, відповідно швидкість редагування,

тестування та налагодження значно зростає. При виявленні помилки інтерпретатор зробить виняток. Наприклад коли програма не опрацьовує виняток інтерпретатор пайтон видає трасування стеку. Також варто сказати, що мова пайтон на рівні вихідного коду дає можливості виконувати код програми покроково з великою кількістю ітерацій, робити перевірку локальних та глобальних змінних.

Варто зазначити, що мова програмування пайтон в порівнянні з іншими розповсюдженими мовами програмування, є більш економічною в плані коду. Що в свою чергу дає зрозумілість коду.

В мові пайтон є основні оператори це:

- арифметичні оператори;
- оператори членства;
- оператори тотожності;
- оператори привласнення;
- побітові оператори;
- логічні оператори;
- цикл for;
- цикл while.

В мові програмування пайтон є класи та об'єкти. Це пов'язано з ООП (об'єктно орієнтоване програмування) – парадигма програмування яка використовується мовою пайтон в якій основними поняттями є поняття об'єктів та класів [11]. Класом можна вважати – групу явищ або предметів, що мають спільні ознаками. Клас в ООП – конструкція мови програмування пайтон, що складається з ключового слова `class`, далі пишеться його наповнення. В кожний клас можна вставляти:

- методи;
- класи;
- поля.

Поля вони ж атрибути визначають стан, а методи визначають поведінку майбутнього об'єкта. Під об'єктом в парадигмі об'єктно-

орієнтованого програмування можна розуміти деяку сутність, що має певний стан і поведінку. Всі методи класу та його поля зберігаються в певній області пам'яті в пулі. Екземпляр класу (instance) описує певний об'єкт у пам'яті, що зберігає у собі нестатичні поля.

Кожний екземпляр класу має свою змінну яка і містить посилання на цей об'єкт. Посилання адреси байта з якого починається об'єкт.

Стан це те як виглядає об'єкт, тобто вміст полів. Поведінка визначає, що саме може робити об'єкт тобто конкретний клас, та визначає методи якими володіє об'єкт.

4.2 HTML

Система повинна мати свій інтерфейс. Зручним у використанні для розробки інтерфейсу є мова гіпертекстової розмітки HTML. Вона надає можливість представляти всі дані та таблиці які були створені у проекті, та може вільно з ним інтегруватися тому для розробки інтерфейсу була вибрана мова HTML.

HTML можна вважати мовою програмування. Основне її завдання це забезпечити розмітку, і використовується вона, щоб вказувати браузеру яким саме чином треба відображати сторінки. Розмітка може бути різної складності, та вона має загальну архітектуру. HTML використовує таке поняття як тег, щоб вкладати або обертати в теги різні частини контенту, це застосовується для того, щоб задати логічну структуру для відображення [12]. Є різні види тегів вони можуть зробити слово або зображення посиланням на щось, можуть зробити слова курсивом, зробити шрифт більше або менше і так далі.

Структура html має вигляд:

```
<!DOCTYPE html>
<html>
```

```

<head>
  <meta charset="utf-8">
  <title> </title>
</head>
<body>
</body>
</html>

```

Де тег `html` включає в себе весь зміст документа тобто туди передадуть всі інші елементи.

Тег `head` призначений для зберігання інших елементів, які повинні забезпечити коректну роботу браузера [13].

Також усередині контейнера `<head>` знаходяться метатеги `meta`, які використовуються для зберігання інформації, призначеної для браузерів і пошукових систем. Як правило при зверненні до них вони допомагають пошуковим системам отримання опису сайту, ключових слів та інших даних.

Тег `body` призначений для збереження вмісту веб-сторінки, що відображається у вікні браузера. Той інтерфейс який треба відобразити повинен розміщуватися саме в цьому тегу. До такої інформації належать текст, теги, скрипти JavaScript тощо.

Елемент `<!DOCTYPE>` призначений для вказівки типу поточного документа фактично це інструкція для кожного документа - DTD (document type definition, опис типу документа) [14]. По суті `<!DOCTYPE>` повинен показати браузеру яким чином треба розуміти, конкретну сторінку, тому що існує багато типів HTML крім того, є XHTML (Extensible HyperText Markup Language, розширена мова розмітки гіпертексту), схожий на HTML, але відрізняється з ним за синтаксисом. По суті він вказує стандарт для оброблення сторінок, спеціальну логіку.

4.3 CSS

Для того, щоб виконати більш якісний інтерфейс системи окрім HTML використовується мова CSS, яка надає можливість зробити оформлення системи за допомогою спеціальних стилів. Тому вона була обрана також як частина інструментів конструювання.

CSS (англ. Cascading Style Sheets – каскадні таблиці стилів) – технологія за допомогою якої оформлюють вигляд документа, оформленого мовою розмітки.

Основний тип застосування як засіб оформлення веб-сторінок у форматі HTML та XHTML, при необхідності може бути використана з іншими документами які мають тип XML [15].

Основні функції які може виконувати CSS:

- завдання кольорів;
- шрифтів;
- розташування;
- фону.

Методологія розробки передбачає що CSS забезпечить поділ вмісту написаного на HTML різних типів, та безпосередньо оформлення документа написаного на CSS. Цей поділ може збільшити доступність документа, завдяки ньому також можна збільшити гнучкість та можливість управління його поданням, дозволяє зроби код більш зрозумілим зменшуєш складність, не вимагає зайвої повторюваності. Крім того, CSS дозволяє представляти той самий документ у різних стилях або методах виведення, таких як екранне представлення, друк.

Таблиці стилів розробляються за певною логікою яка робить роботу коректною.

Загальний вигляд CSS запису такий:

selector {property: value} .

Де селектор (Selector) – це елемент, до якого будуть застосовуватися стилі, що призначаються. Це може бути тег, клас або ідентифікатор гіпертекстового об'єкта документа.

Властивість (Property) визначає одну чи кілька характеристик селектора. Властивості задають формат відображення селектора: відступи, шрифти, вирівнювання, розміри тощо.

Значення (Value) – це фактичні числові чи рядкові константи, що визначають властивість селектора.

Опис (Declaration). Сукупність властивостей та їх значень.

Правило (Rule). Повний опис стилю (селектор + опис).

CSS дозволяє регулювати розмір елементів.

Height встановлює висоту елемента. Може набувати значення auto, визначатися абсолютною або відносною (у відсотках) величиною.

Line-height встановлює відстань між лініями. Може приймати значення normal, number, визначатися абсолютною чи відносною (у відсотках) величиною.

Max-height встановлює максимальну висоту елемента. Може набувати значення none, визначатися абсолютною або відносною (у відсотках) величиною.

Max-width встановлює максимальну ширину елемента. Може набувати значення none, визначатися абсолютною або відносною (у відсотках) величиною.

Min-height встановлює мінімальну висоту елемента. Може набувати значення none, визначатися абсолютною або відносною (у відсотках) величиною.

Min-width встановлює мінімальну ширину елемента. Може набувати значення none, визначатися абсолютною або відносною (у відсотках) величиною.

Width встановлює ширину елемента. Може набувати значення auto, визначатися абсолютною або відносною (у відсотках) величиною.

Left, top, right, bottom дозволяють робити відступи відповідно право, ліво, зверху, знизу.

Однією з важливих властивостей CSS є можливість позиціонування блочних елементів за допомогою.

Static – нормальне позиціонування цей блок є звичайним блоком, він відображається відповідно до загальних правил. Властивості 'left' та 'top' не застосовуються.

Relative – відносне позиціонування положення блоку розраховується відповідно до свого нормального (static) положення.

Absolute – абсолютне позиціонування положення блоку (можливо і розмір) вказується за допомогою властивостей 'left', 'right', 'top' та 'bottom'. Вони вказують величину пересування відносно контейнера блоку. Абсолютно блоки, що позиціонуються, вилучаються з нормального потоку. Тобто вони змінюють послідовність блоків які були з ними.

Fixed – фіксоване положення блоку розраховується відповідно до моделі абсолютного позиціонування, а потім він фіксується щодо області перегляду або сторінки.

Загалом були наведені лише деякі властивості CSS спектр яких набагато ширший.

4.4 Інтегроване середовище розробки pycharm

Для якісного написання коду на Python який вимагає наша система, треба використовувати спеціальне програмне середовище. Воно повинно вміти добре працювати з синтаксисом мови на якій пишеться код, робити компіляції коду, за необхідності показувати помилки, мати вбудовані пакети програмні. Цим критеріям відповідає PyCharm тому він і був обраний для розробки.

IDE (інтегроване середовище розробки) – спеціальний редактор для обробки програмного коду, не варто його плутати з текстовим редактором

Зазвичай поширені текстові редактори для коду, такі як Sublime або Atom, можуть запропонувати багато зручних функцій, таких як підсвічування синтаксису, інтерфейс який можна налаштовувати і розширені засоби навігації, що дозволяють тільки писати код. Для створення функціонуючих програм, як мінімум, потрібен компілятор і відладчик [16].

IDE на відміну від звичайного редактору коду включає ці компоненти, як і багато інших. Також можливо постачання додаткових пакетів з них и для автоматизації, тестування та візуалізації процесу розробки. Сам термін «інтегроване середовище розробки» означає, що дане середовище має все необхідне для перетворення написаного коду на функціонуючі програми.

Основними властивостями pycharm є допомога під час редагування.

Розумний редактор PyCharm призначений для максимально продуктивної розробки на Python, JavaScript, CoffeeScript, TypeScript, CSS та популярних мовах шаблонів. Функції автодоповнення, виявлення помилок та швидкі виправлення враховують особливості кожної з мов, що підтримуються.

Налагодження, тестування та профілювання.

Данна IDE надає можливість використовувати власний відладчик для Python та JavaScript. Забезпечує запуск тестів, при цьому він дозволяє використовувати функції розумного редагування, також є можливість переглянути звіти про запуск у зручному графічному інтерфейсі [17].

Варто зазначити що дане інтегроване середовище розробки має інтерактивну консоль для Python. Можливо здійснити запуск консолі REPL для Python, яка має багато переваг над стандартною консоллю. Серед них перевірка синтаксису на льоту за допомогою інспекцій, зіставлення дужок та лапок та, звичайно, автодоповнення.

До переваг даної платформи входить також зручний інтерфейс який можна налаштовувати.

Цей інтерфейс побудований так для даної IDE, що може максимально адаптуватися до потреб користувача. Тобто він має можливість налаштувати

робоче середовище так, як подобається, вибрати відповідну колірну схему та зручні поєднання клавіш.

4.5 Архітектура MVT

Для зручного написання коду та логіки системи треба використовувати модель яка б підійшла до платформи django. Вона повинна забезпечити чітке функціональне розділення для проєктованих вузлів, та повинна забезпечити їх інтеграцію. Саме це може зробити модель MVT. Тому саме вона була вибрана для розробки дипломної системи.

Концепція Django заохочує вільне зв'язування та суворий поділ частин програми. Якщо слідувати цій філософії, то легко вносити зміни в одну конкретну частину програми без шкоди для інших частин. У функціях представлення, наприклад, дуже важливо відокремлення бізнес-логіки від логіки відображення за допомогою шаблонної системи. Використовуючи шар для роботи з базою даних, ми застосовуємо цю філософію для логіки доступу до даних [18].

Основу моделі MTV створюють три компоненти – логіка доступу до даних, бізнес-логіка та логіка відображення – становлять концепцію, яку називають Модель-Представлення-Шаблон (Model-View-Template, MVT) архітектури програмного забезпечення. У цій концепції термін "Модель" відноситься до логіки доступу до даних; термін «Уявлення» відноситься до тієї частини системи, яка визначає, що показати і як; а термін «Шаблон» відноситься до тієї частини системи, яка визначає яке уявлення треба використовувати, залежно від введення користувача, за необхідності отримуючи доступ до моделі.

М визначають як "Модель" (Model), тобто це шар доступу до даних. Цей шар знає все про дані: як отримати доступ до них, як перевірити їх, як з ними працювати і як дані пов'язані між собою.

T визначають як Шаблон (Template), це спеціальний шар представлення даних. Цей шар приймає рішення щодо представлення даних: як і що має відображатися на сторінці або іншому типі документа.

V визначають як "Уявлення" (View), шару бізнес-логіки. Цей шар містить логіку, як отримати доступ до моделей і застосовувати відповідний шаблон. Його можна розглядати як зв'язок між моделями та шаблонами.

Роботу моделі MTV зображено на рисунку 4.1.

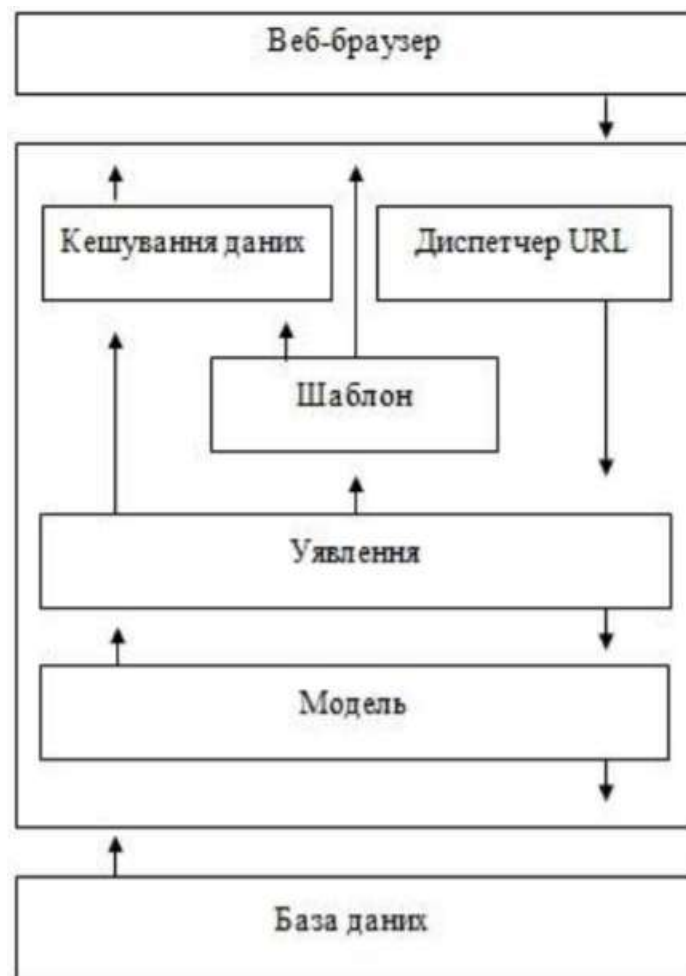


Рисунок 4.1 – Робота моделі MTV [19]

4.6 Проектування медичної інформаційної інфраструктури

4.6.1 Вимоги до системи медичної інформаційної інфраструктури

Для початку роботи нам треба визначити структуру вимог які ставляться до розроблюваної медичної інформаційної інфраструктури.

Бізнес вимоги.

Дана система повинна забезпечити зручну, комфортну систему запису до лікарів, лікарі повинні мати змогу визначати графік прийому, самостійно його корегувати, та переглядати усі записи до себе. Також дана система повинна мати великий ступінь модульності, за необхідності легко приєднувати нові модулі, не перекриваючи функціонал вже існуючих.

Дана система повинна буди автономною і мати змогу працювати як з окремими установами ,так із цілими комплексами.

Система повинна бути зрозумілою в користуванні, її модулі повинні буди максимально автономними і мати змогу діяти як в системі, так і відокремлено від неї, як самостійні одиниці.

Вимоги користувачів.

Система повинна забезпечити зручний спосіб комунікації та запису на прийом до лікаря, мати зрозумілий функціонал.

Система повинна надавати можливість переглядати особистий кабінет користувача, дивитися записи користувача до лікаря.

Також повинна бути функція статистики. щоб користувач міг переглядати використані препарати, та до яких діагнозів вони застосовуються.

Функціональні вимоги.

Система повинна мати функцію запису до лікаря, відміни запису до лікаря ,перегляд статистики використаних препаратів, функції налаштування та зміни графіку для лікарів, функцію призначення препаратів

4.6.2 Структура бази даних медичної інформаційної інфраструктури

Проект даний передбачає використання даних для різних цілей, дані повинні мати свою ієрархію.

Причому, треба враховувати те, що данні повинні між собою сполучатися, зв'язуватися. Потрібно в одну таблицю викликати дані з іншої таблиці.

Тому для цього ми використаємо реляційні бази даних. Саме вони забезпечують зв'язаність багатьох даних.

У реляційних базах даних вся інформація міститься в таблиці. Стовпці у такій таблиці мають конкретний тип та розмір; весь вміст ячейки стовпця має відповідати їх типу. Рядки є набір неструктурованих даних і утворюють математичне відношення. Таким чином, вся база даних – це безліч таблиць-відносин, звідси і назва моделі (від англ. Relation – відношення). Причому такі таблиці можуть поєднуватися за допомогою спеціальних зв'язків

Один до одного це коли кожному запису першої сутності відповідає лише один запис із другої сутності. А кожному запису другої сутності відповідає лише один запис із першої сутності [20].

Один до багатьох коли кожен запис першої сутності може відповідати кільком записам з другої сутності. Однак кожному запису другої сутності відповідає лише один запис із першої сутності. Наприклад, є дві сутності: Замовлення та позиція замовлення. І в одному замовленні може бути багато товарів [21].

В відношенні багато до багатьох кожен запис першої сутності може відповідати кільком записам з другої сутності. Однак і кожного запису другої сутності може відповідати кілька записів першої сутності. Наприклад, є дві сутності: Автор та Книга. Один автор може написати багато книжок. Але книга може мати кілька авторів [22].

За критерієм обов'язковості відносини поділяються на обов'язкові та необов'язкові.

Обов'язкове відношення означає, що з кожного запису з першої сутності обов'язково повинні бути присутні пов'язані записи на другий сутності.

Необов'язкове відношення означає, що записи з першої сутності може й не існувати у записі другої сутності.

Наразі ми створимо сутність для реєстрації. Тип зв'язку будемо використовувати один до багатьох тобто наша сутність включатиме в себе багато інших сутностей.

Основний тип зв'язку який буде використовуватися в таблицях це один до багатьох, саме такий тип дозволить оптимально з'єднувати таблиці.

За допомогою таблиці Med таблиця 1.4 буде створюватися сутність мед працівника.

Таблиця 4.1 – Таблиця Med

Ім'я	Тип даних
Name	CharField
Id	Integer

За допомогою таблиці Month буде створюватися сукупність місяців таблиця 4.2.

Таблиця 4.2 – Таблиця Month

Ім'я	Тип даних
Month	CharField
Id	Integer

За допомогою таблиці Chis буде створюватися сукупність чисел
таблиця 4.3.

Таблиця 4.3 – Таблиця Chis

Ім'я	Тип даних
Chis	CharField
Id	integer

За допомогою таблиці Cifri буде створюватися сукупність часу (табл.
4.4.).

Таблиця 4.4 – Таблиця Cifri

Ім'я	Тип даних
Cifri	CharField
Id	integer

Наразі створили сукупність таблиць які забезпечать формування
запису до лікаря для всіх сутностей медичної інформаційної системи.

Далі необхідно створити таблиці для конкретного доктора, за
допомогою яких буде визначатися його конкретний графік.

Таблиця 4.5 забезпечить вибір робочого місяця конкретного лікаря.

Таблиця 4.5 – Таблиця MonthA

Ім'я	Тип даних
month	ForeignKey
Id	integer

Таблиця 4.6 забезпечить вибір робочих днів для обраного місяця
конкретного лікаря.

Таблиця 4.6 – Таблиця ChisA

Ім'я	Тип даних
Chis	ForeignKey
Id	integer

Таблиця 4.7 забезпечить вибір робочого часу для робочих днів для обраного місяця конкретного лікаря.

Таблиця 4.7 – Таблиця CifriA

Ім'я	Тип даних
Cifri	ForeignKey
Id	integer

Після створення всіх основних таблиць за допомогою яких будуть формувати базу для запису до лікаря, їх треба об'єднати в одну велику таблицю яка буде дозволяти вибирати записи з інших (табл. 4.8).

Таблиця 4.8 – Таблиця grafA

Ім'я	Тип даних
Month	ForeignKey
ChisA	ForeignKey
grafa	ForeignKey
DNAME	ForeignKey
Ures	ForeignKey
Id	ForeignKey

Наступним кроком треба створити систему обліку препаратів та призначень. Для створення данної системи будуть створені такі таблиці.

Таблиця 4.9 буде записуватися загальні діагнози, які будуть використовуватися для при вироблені загального напрямлення.

Таблиця 4.9 – Таблиця Diagnoz1

Ім'я	Тип даних
Diagnoz	CharField
Id	integer

Таблиця 4.10 створення для занесення всієї препаратної бази, які будуть призначатися.

Таблиця 4.10 – Таблиця Preparat1

Ім'я	Тип даних
Preparat	CharField
Id	Integer

Таблиця 4.11 це загальна таблиця в якій кожному діагнозу призначають препарат, також там вказаний лікар, та відвідувач. Таблиця зв'язується з попередніми по ForeignKey.

Таблиця 4.11 – Таблиця statistical

grafa	ForeignKey
Doctor123	ForeignKey
diagnoz1	ForeignKey
preparat1	ForeignKey
us_zapis	ForeignKey
Id	Integer

На рисунку 4.2 зображена схема взаємодії таблиць бази даних які загалом створюють модуль запису до лікаря.

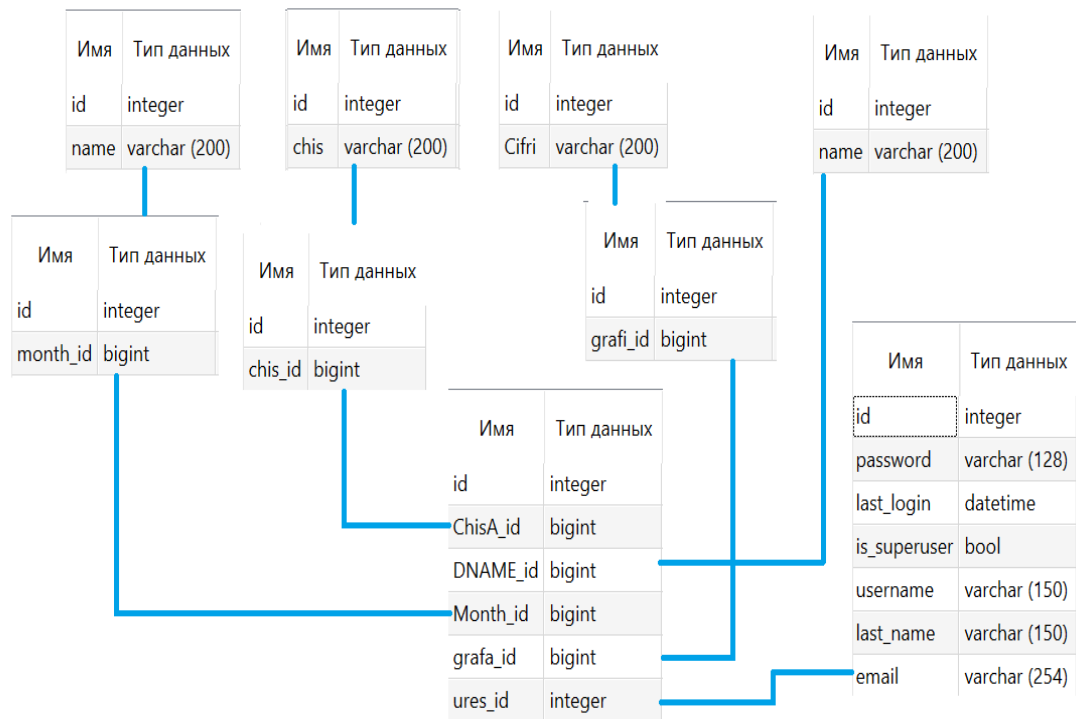


Рисунок 4.2 – Структура бази даних реєстрації

Як можна побачити на рисунку 4.2 всі таблиці які утворюють таблицю grafA усі вони об'єднанні зв'язком.

На рисунку 4.3 зображена схема організації бази даних для функції рецепту один до багатьох.

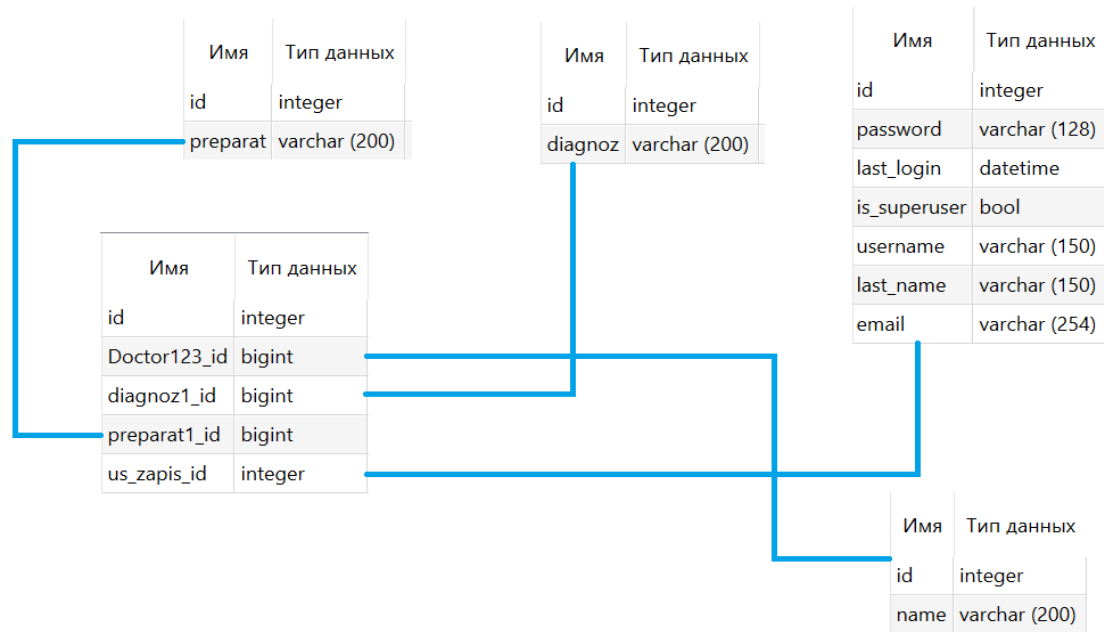


Рисунок 4.3 – Структура бази даних рецепту

Як і таблиця 4.2 таблиця 4.3 теж включає в себе всі попередньо створені таблиці за принципом один до багатьох.

4.6.3 Програмне забезпечення функціонування медичної інформаційної інфраструктури

Для того щоб база даних функціонувала її треба створити програмно. В джанго для цього передбачений спеціальний файл який називається models. Таблиці в базі даних в джанго представлені за допомогою класів. Вони позначаються як class.

Всі таблиці в базі даних створюють на основі базового класу models. Перед початком роботи його треба імпортувати це виглядає наступним чином:

```
from django.db import models .
```

Після того як був імпортований базовий клас необхідно унаслідувати його в створюваній таблиці, таблиця буде створюватися наступним чином:

```
class Med(models.Model):
    name = models.CharField(max_length=200,verbose_name='Медпрац')
    def __str__(self):
        return self.name
    class Meta:
        verbose_name = 'Працівник'
        verbose_name_plural = 'Працівник'
```

По аналогії будуть створюватися всі інші частини бази даних.

Де `models.Model` унаслідування базового класу. Назва `name` після треба вказати тип використовуваних даних в таблиці `models.CharField`.

Також треба задати параметри які будуть характеризувати таблицю `max_length` можна вказати кількість символів, `verbose_name='Медпрац'` назва для відображення назви строки.

Наступним чином нам треба відображати нашу фрму для цього треба з'єднати модель яку створили та форму.

Створимо модель для запису до лікаря.

```
class grafA(models.Model):
    Month = models.ForeignKey('MonthA', on_delete=models.CASCADE,
        default=1,verbose_name='Місяць' )
    ChisA = models.ForeignKey('ChisA', on_delete=models.CASCADE,
        verbose_name='Число' )
    grafA = models.ForeignKey('CifriA', on_delete=models.CASCADE,
        verbose_name='Тайм' )
    DNAME = models.ForeignKey('Med', on_delete=models.PROTECT,
        default=1, )
```

```

        ures = models.ForeignKey(User, on_delete=models.PROTECT,
null=True, verbose_name='клиент')

```

```

def __str__(self):
    return str(self.Month)

```

Дана таблиця відрізняється від попередньої тим, що вона до себе включає інші таблиці.

Де `models.ForeignKey` означає тип даних який показує, що дане поле використовує зв'язок один до багатьох.

Приччому 'Med' вказує яка саме таблиця прив'язується до даної, де `on_delete=models.PROTECT`, інструкція яка вказує алгоритм роботи при зміні даних записів.

Наступним кроком створимо форму яка дозволить відобразити модель.

Форми створюватимуться за допомогою базового класу `forms`, від якого вони всі унаслідуються. Для цього його імпортують:

```

from django import forms .

```

Сама форма представляє собою також клас як і сама модель.

```

class grafForms(forms.ModelForm):
    class Meta:
        model= grafA
        fields=['Month','ChisA','grafa']
        Month = forms.CharField(label="Місяць")
        ChisA = forms.CharField(label="Число")
        grafA = forms.CharField(label="Час")

```

Де grafForms назва форми. Клас який унаслідували forms.ModelForm.

Також ,щоб форма чітко відображала модель її треба теж унаслідувати model= grafA де grafA унаслідувана модель створена вище.

Формі треба вказати полі які вона буде показувати fields=['Month','ChisA','grafa'] тут 'Month','ChisA','grafa' означають назви полів.

Також кожному полю треба задати його назву це робить за допомогою:

```
Month = forms.CharField(label="Місяць");
ChisA = forms.CharField(label="Число");
grafa = forms.CharField(label="Час").
```

Де назви полів в таблиці:

- Month;
- ChisA;
- grafa.

Наступним завданням треба створити функції які зможуть оброблювати моделі.

Так як ми, будемо створювати форму яка буде реєструвати створення нових полів в базі даних ми повинні використовувати спеціальні клас які для цього є в джанго. Клас CreateView його треба імпортувати.

```
from django.views.generic import CreateView
```

Далі треба треба створити сам клас який буде працювати з моделлю.

```
class Registration_doctor(CreateView):
    form_class=grafForms
    template_name = 'DPU/register_doc.html'
```

```

success_url = reverse_lazy('vvr')

def form_valid(self, form):
    dz=form.cleaned_data
    form.instance.ures = self.request.user

    if grafA.objects.filter(Month=dz['Month'], ChisA=dz['ChisA'],
    grafa=dz['grafa']).count()==0:
        return super(Registration_doctor, self).form_valid(form)
    else:
        pass
        #return redirect("")

```

Де в класі треба вказати форму яку ми наслідуємо
`form_class=grafForms`.

Наступним кроком вказати шаблон у якому оброблятиметься форма
`template_name = 'DPU/register_doc.html'`.

Також можна вказати куде перенаправляти користувача якщо
 відправка форми відбулась коректно `success_url = reverse_lazy('vvr')`.

Наступним завданням треба створити функцію яка буде контролювати
 данні які вводить користувач і перевіряти їх на наявність такого ж запису в
 базі даних. Це треба, щоб у одного лікаря не було двох однакових записів.

Ця функція має вигляд `def form_valid(self, form)` де:

```
dz=form.cleaned_data ,
```

представляє собою змінну `dz` в яку присвоєно метод `cleaned_data`

Суть методу `cleaned_data` полягає в тому, що він передає лише
 перевірені, тобто коректно введені дані які ввів користувач.

В системи користувачів багато і кожного разу коли робиться запис до бази даних треба вказувати користувача, можна зробити так, щоб користувач вибирав себе із списку, але це робить запис не дуже зручним.

Тому для того ,щоб користувач автоматично заповнювався використовується функція `form.instance.user = self.request.user`.

Для того, щоб пройшла перевірка на наявність запису з такими даним які ввів користувач у форму треба зробити спеціальну умову:

```
if grafA.objects.filter(Month=dz['Month'], ChisA=dz['ChisA'],
    grafa=dz['grafa']).count()==0
```

Де `grafA.objects.filter` значить, що йде звернення до бази даних `grafA` діла обираємо об'єкти з неї за допомогою `objects` та встановлюємо фільтр `filter`.

У фільтрі пишемо умову де `Month=dz['Month']` значить, що поле в таблиці `grafA` з назвою `Month` буде дорівнювати, даним які надійшли від користувача та пройшли перевірку `dz['Month']` `Month` це рядок який заповнив користувач.

Далі треба зазначити, що кількість записів які відповідають умові фільтру `Month=dz['Month']` , `ChisA=dz['ChisA']`, `grafa=dz['grafa']` повинні відповідати кількості записів `count()==0`.

Це означатиме, що дані які вводить користувач є унікальними та вони не повторюються в базі даних.

Після того як дані пройшли перевірку вони повинні назад повернутися це забезпечується за допомогою `return`. Вінт повертає форму `return super(Registration_doctor, self).form_valid(form)` .

І форма зберігається в базі даних.

Наступним кроком треба зробити так . щоб користувач міг звертатися до бази даних та робити записи до неї.

Для цього створено спеціальний обробник url адрес .

Для його створення треба імпортувати модуль `path`.

```
from django.urls import path
```

Після імпортування необхідно прописати путь який буде оброблювати введену користувачем юрл адресу.

```
path('hkkn', Registration_doctor.as_view(),name=' hkkn ')
```

Де 'hkkn' назва url адреси яку користувач повинен вводити в браузері, але під час розробки на даний юрл може посилатися багато функцій та класі, і можливо таке, що його доведеться змінити відповідно його доведеться змінити і у всіх модулях де він використовувався. Для убезпечення вводять name=' hkkn ' це фактично назва самої url адреси і вони можуть співпадати.

Тепер при зміні самого юрл не зміниться його назва, що дозволить його зручно використовувати.

Але при зверненні до юрл адреси повинні активізуватися функції, які будуть оброблювати моделі для запису та створення запитів.

Тому кожній юрл адресі треба присвоїти функцію, або спеціальний клас для створення записів.

Якщо ми створюємо клас ми повинні вказати його назву Registration_doctor, та викликати спеціальний метод as_view().

Після створення запису в таблиці реєстрації користувач, повинен бачити на який час він записаний.

Для цього використовують спеціальну функцію:

```
def user_check_list(request,):
    context = {'Podstanovka': grafA.objects.filter(ures=request.user)}
    return render(request, 'DPU/v_VR.html', context ),
```

Де def значит, що запис функція. Також функція повинна передавати якісь данні context = {'Podstanovka': grafA.objects.filter(ures=request.user)}.В

даному випадку ми передаємо словник в якому передаємо данні з таблиці grafA, обираємо об'єкти .objects, вказуємо фільтр filter(ures=request.user) який вибирає записи які створив користувач.

Також у медичній інформаційній інфраструктурі передбачено, що лікарі зможуть самі вибирати собі графік.

Для цього створюється спеціальний клас який буде оброблювати форму для запису графіку лікаря яка робиться від моділе про яку вказано вище.

```
class Doctor_list_Month( CreateView):
    form_class=MonthForms
    template_name = 'DPU/Month.html'
    success_url = reverse_lazy('vvr')
```

Даний клас є дочірнім класов CreateView, який дозволяє створювати записи в базі даних.

Де form_class=MonthForms задає форму з якою працює клас.

Зазначає який шаблон використовувати template_name = 'DPU/Month.html'.

Так само кожний клас повинен оброблюватися обробником url адрес.

В даному випадку треба вказати назву юрл адресу, при потребі вказати

```
path('dl', Doctor_list_Month.as_view(),name='dl').
```

Також у системі функціонує можливість перегляду статистики, для чого розроблена спеціальна функція.

```
def naz(request):
    nn=statistica1.objects.filter(preparat1=1).count()
```

```

context = {'Podstanovka': statistica1.objects.filter(preparat1=21), 'bb':nn
}

return render(request, 'DPU/Otobrazhenie1.html', context )

```

Дана функція повинна відобразити користувачу всі записи обраного препарату, та показати до яких діагнозів він застосовується.

В даній функції в словник передані записи з таблиці `statistica1` з неї обрано тільки ті препарати у яких `preparat1=1`, тобто вони мають ключ номер 1.

Також вони повинні показати кількість використаних препаратів за допомогою `nn=statistica1.objects.filter(preparat1=1).count()`.

Для більшої інформативності також треба показувати статистику діагнозу це робиться за допомогою спеціальної функції яка покаже які препарати застосовувалися до конкретного діагнозу.

```

def preparat(request):
    nn=statistica1.objects.filter(preparat1=1).count()
    context = {'Podstanovka': statistica1.objects.filter(preparat1=21), 'bb':nn
}

return render(request, 'DPU/Otobrazhenie.html', context )

```

Діє дана функція аналогічним порядком як і попередня вона в словник передає звернення до бази даних `statistica1.objects` включає фільтр `preparat1=21`, який обере записи з бази даних `statistica1` ті у яких `diagnoz1=21`.

Далі так само як і попередня функція визначить за допомогою `nn=statistica1.objects.filter(preparat1=1).count()` кількість таких записів.

Також функція має повернути `return render(request, 'DPU/Otobrazhenie.html', context )`, контекст і те де вона його буде показувати.

4.6.4 Робота системи

У попередньому розділі була описана програмна частина роботи системи та її принципи всі інші частини медичної інформаційної системи були створені таким саме чином. І функціонують вони за одним принципом. Весь інший код наведений в додатку А.

Охарактеризувати роботу системи можна таким чином. На самому початку було створено чотири основні мета таблиці даних, їх призначення надавати можливість лікарям обирати свій робочий графік.

Також однією із цих таблиць є та, що створює сутності лікарів.

Вони повинні взаємодіяти з усіма локальними таблицями для кожного лікаря і надавати необхідні можливості для створення графіку.

Далі були створені спеціальні таблиці які повинні належати конкретному лікарю, де він безпосередньо визначає свій графік.

Причому мета таблиці повинні бути зв'язані з локальними, бо саме вони дають змогу чітко та регламентовано формувати графіки лікарів.

Далі в системі є модулі які призначені для реєстрації користувача, та його аутентифікації.

Таблиці локального рівня для конкретного лікаря, та таблиці які визначають користувача, та таблиця для визначення лікаря, за допомогою ключа один до багатьох об'єднуються у систему реєстрації користувача до лікаря.

Наступним елементом системи є функція яка забезпечує перегляд користувачу записів які він зробив до лікаря, з вказанням всіх даних які заповнив користувач.

Також в системі діє функція відміни запису до лікаря.

Наступним елементом системи є функціонал перегляду статистики використаних препаратів до діагнозу, та діагнозів до препарату.

Вона реалізовується за допомогою створення таких таблиць як діагноз де встановлюється необхідна та регламентована їх кількість.

Далі створюється спільна таблиця, де власне і відбувається призначення препарату до діагнозу.

Ця таблиця включає в себе таблиці користувачів, лікарів, діагнозів та препаратів. Вони об'єднуються за допомогою зв'язку один до багатьох.

Далі було організовано функцію ,яка дозволяє дивитися статистику, для ліків обрали якийсь засіб і подивилися до яких діагнозів застосовується, та обрали діагноз і подивилися які ліки застосовуються.

4.6.5 Логіка взаємодії користувача з системою

Спілкування клієнта із сервером відбувається за протоколом HTTP. Основою цього протоколу є запит від клієнта до сервера та відповідь сервера клієнту.

Для запитів в данній системі як правило використовують методи GET, якщо хочемо отримати дані, і POST, для того щоб можна було змінити дані. У самому запиті вказується Host (домен сайту), також вказується тіло запиту (якщо це POST-запит).

На даному етапі основним протоколом для спілкування є HTTPS, фактично це розширена версія HTTP із підтримкою шифрування SSL/TLS. Тобто шифрований канал передачі даних використовується за замовчуванням незалежно від типу інформації яку передали.

Перед самим HTTP також використовують запит DNS (domain name system). За допомогою цього запиту можна отримати необхідну ip-адресу.

Коли запит від браузера доходить до сервера, він не одразу потрапляє до джанго. Найперше його повинен обробити веб сервер Nginx. Варто зауважити, що він відправляє виключно статичні файли. Якщо запит не до статички, то Nginx має передати його до джанго.

Nginx може передати цей запит за допомогою іншої програми. Тому використовується ще одна програма-прошарок-сервер додатків. WSGI. Веб сервер приймає та обслуговує запити. Він може віддавати файли (HTML,

зображення, CSS, і т.д.) безпосередньо зі своєї файлової системи. Однак він не може напряду працювати з Django застосуванням. WSGI необхідний інтерфейс, який буде запускати Django застосування, передавати йому запити від веб клієнта, наприклад браузера, і повертати відповіді [23].

На головній сторінці показані лікарі, та виведені посилання на запис до них рисунок 4.4.

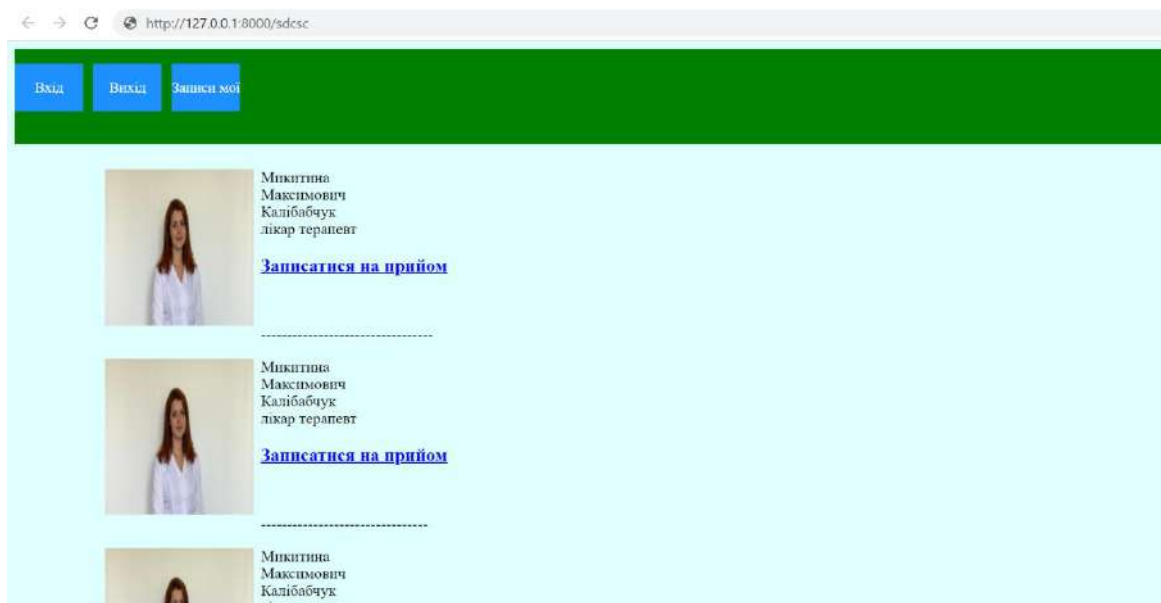


Рисунок 4.4 – Головна сторінка медичної інформаційної системи

Далі розроблено форму для запису до лікаря з можливістю обирати час дату, місяць рисунок 4.5.

The screenshot shows a web browser window with the address bar displaying `http://127.0.0.1:8000/hkkn`. The page has a green header bar with three buttons: "Вхід", "Вихід", and "Записи мої". Below the header, on a light blue background, there is a form for making an appointment. The form consists of three stacked input fields with labels "Оберіть місяць", "Оберіть дату", and "Оберіть час". Below these fields is a button labeled "Записатися".

Рисунок 4.5 – Інтерфейс запису до лікаря

На рисунку 4.6 зображено кабінет користувача де він може переглянути свої записи та подивитися статистику рисунок 4.6.

The screenshot shows a web browser window with the address bar displaying `127.0.0.1:8000/check_list`. The page has a green header bar with three buttons: "Вхід", "Вихід", and "Записи мої". Below the header, on a light blue background, there is a section titled "Ви записані до лікаря:". Under this title, the following information is displayed: "Місяць: Грудень", "Дата: 1", "Час: 10", and "Ваш лікар: Микитина Максимович Калібабчук". Below this information is a red link labeled "Відмінити запис до лікаря". To the right of this information is a blue circular button labeled "Назад". Further to the right, there are two rectangular buttons: "Оберіть препарат та перегляньте статистику його використання" and "Оберіть діагноз та перегляньте статистику використання препаратів для нього".

Рисунок 4.6 – особистий кабінет користувача

На рисунку 4.6 можна побачити меню для перегляду статистика використання препарат.

Була розроблена функція медичної інформаційної системи при якій користувач може вибрати необхідний йому діагноз та подивитися статистику використання препаратів (рис. 4.7). Побачити до яких напрямлень він застосовується.

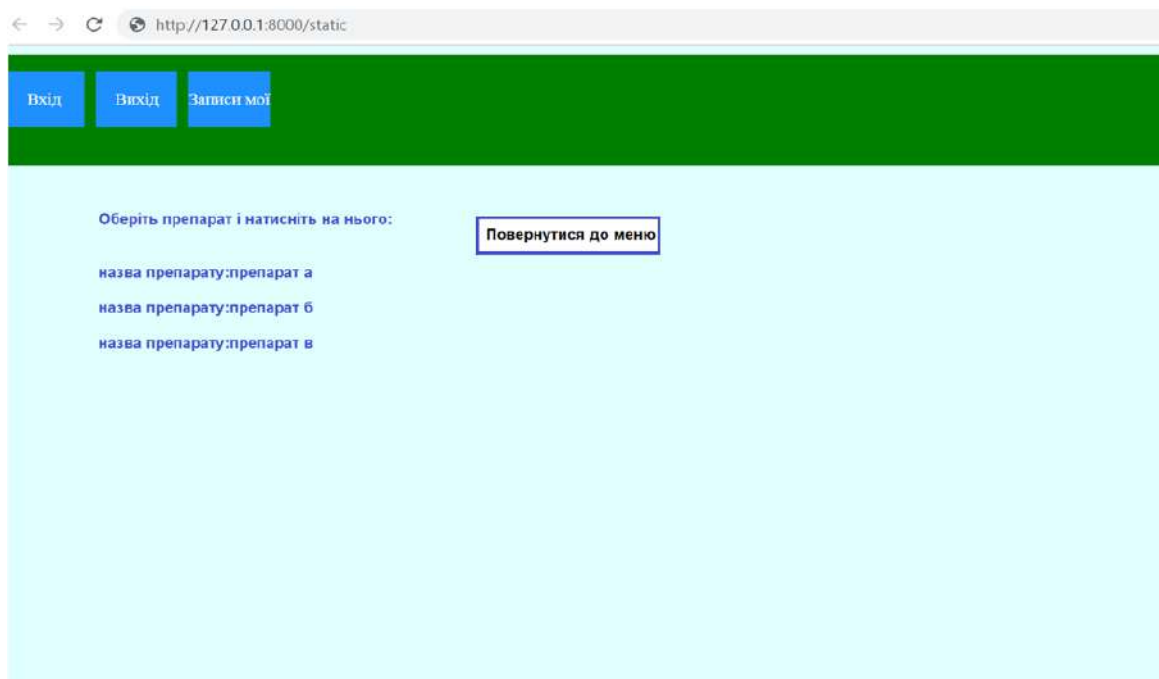


Рисунок 4.7 – Вивід меню статистики

На рисунку 4.8 зображено вивід статистичних даних конкретного діагнозу, «Діагноз а» та використаних препаратів до нього.

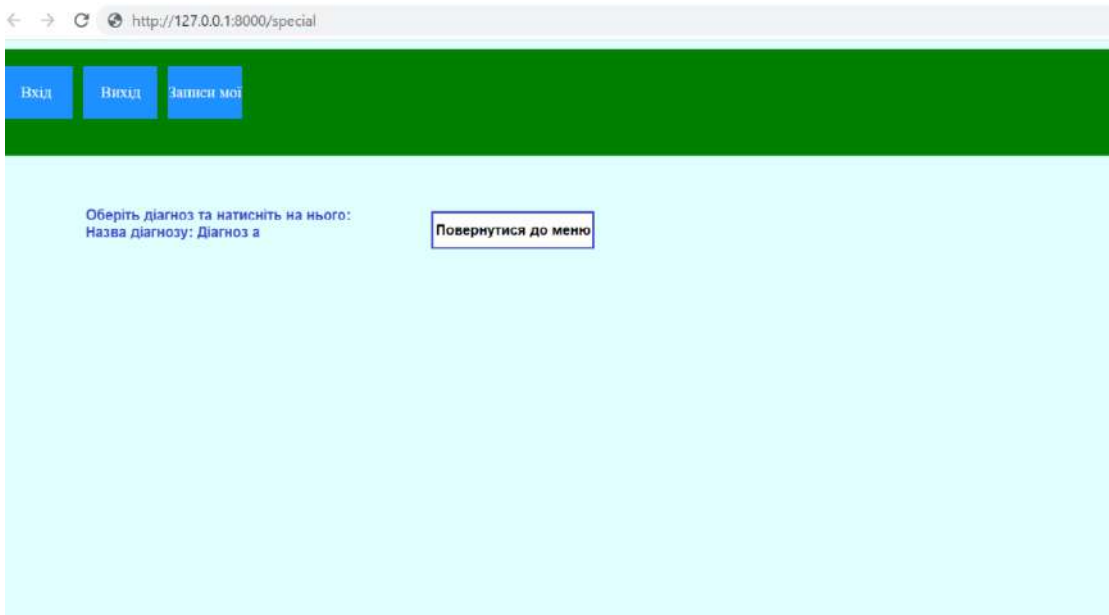


Рисунок 4.8 – Вивід меню статистики

На рисунку 4.9 зображено вивід статистики.

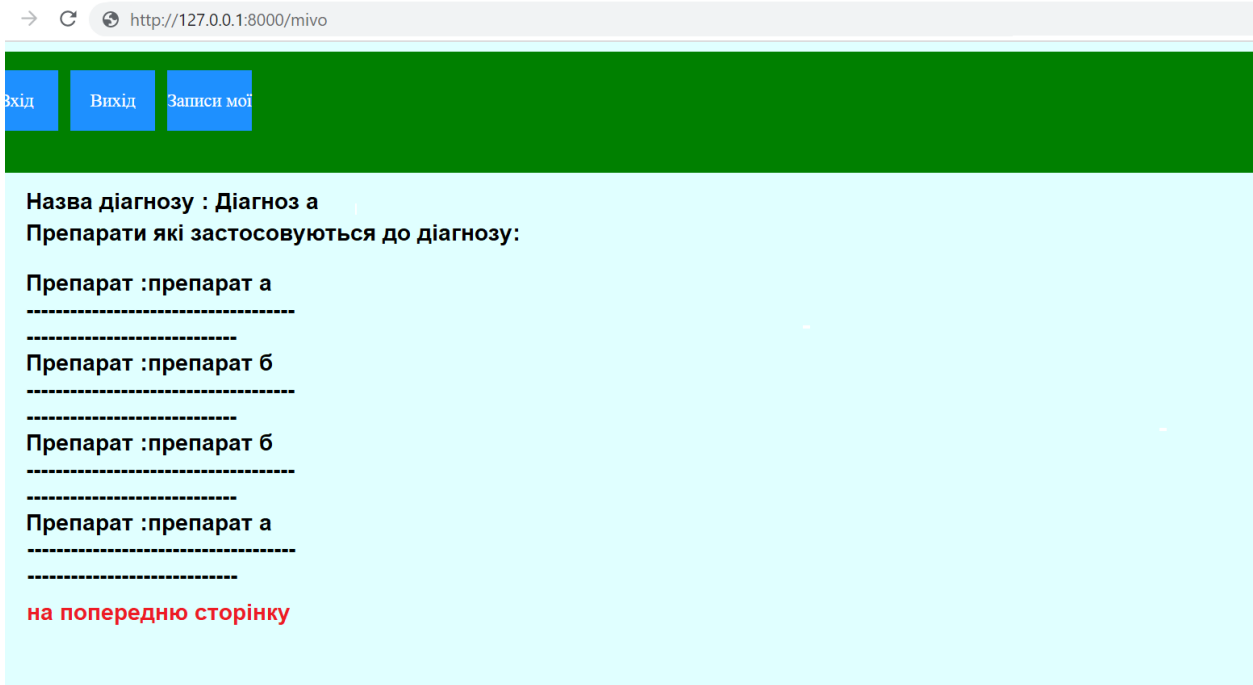


Рисунок 4.9 – Вивід статистики

Данна статистика може давати великі можливості для аналітики використання препаратів до діагнозу до, аналізу їх можливостей. Вона розширює можливості як для лікаря та для пацієнта теоретично зможе покращити використання препаратів. Виявити їх взаємодію з різними типами діагнозів.

Також передбачена функція виводу використання до конкретного препарату та діагнозів до яких даний препарат застосовується, дана статистика полегшує роботу лікарю з аналізу ефективності призначення препарату, дозволяє визначити його властивості, та інші функції необхідні для використання, що демонструє рисунок 4.10.

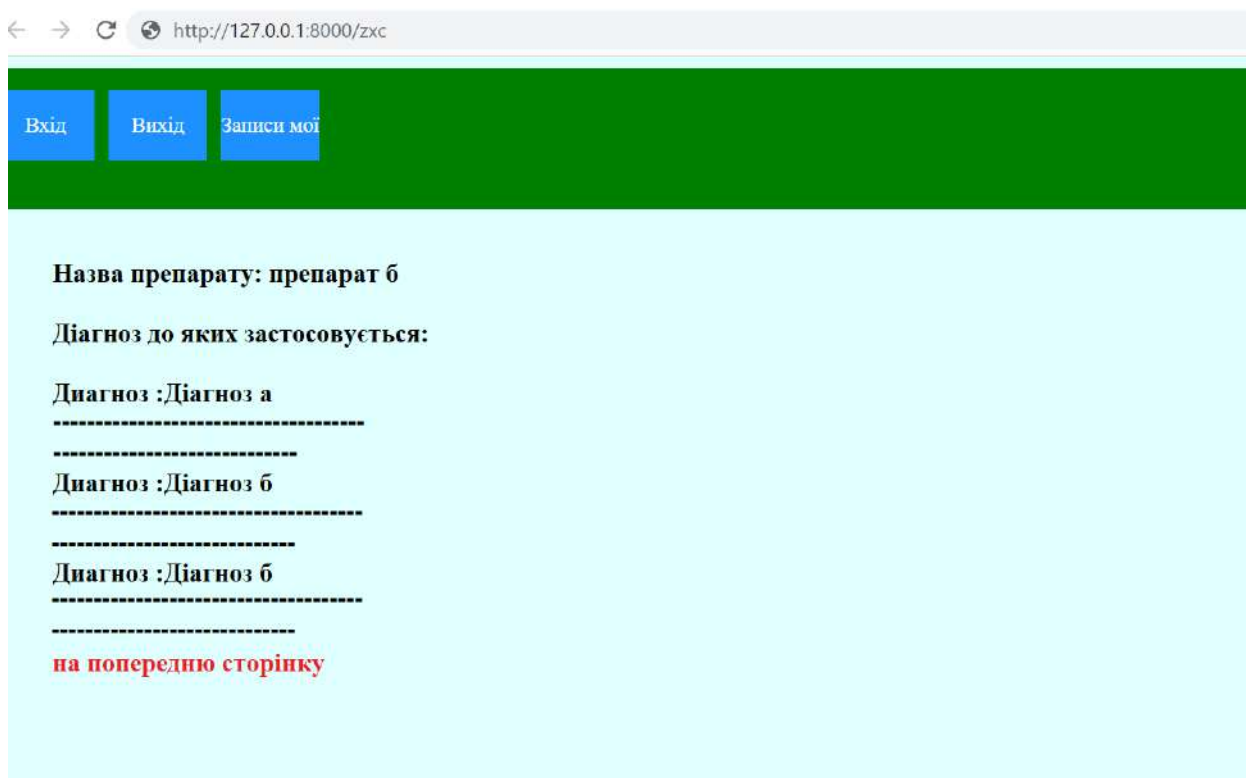


Рисунок 4.10 – Вивід статистики

Даними скріншотами була представлена лише основна частина функціоналу системи, також варто зазначити, що

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Опис ідеї медичної інформаційної інфраструктури

В Україні у медичних організаціях запроваджено медичну електронну інформаційну інфраструктуру, електронні системи допомоги у прийнятті рішень, створено особисті кабінети пацієнта, організовано робоче місце лікаря та медичної сестри, забезпечено доступ до електронних інформаційних та навчальних ресурсів.

Наразі розробляються та впроваджуються системи загальнодержавного значення, які могли б забезпечити дистанційне використання, людиною функцій які вона раніше робила безпосередньо у паперовому вигляді, а тепер зможе в електронному (табл. 5.1).

Таблиця 5.1 – Опис ідеї

Зміст ідеї	Напрямки застосування	Вигоди для споживачів (користувачів)
Розробка медичної інформаційної інфраструктури приватної медичної установи. Дозволи покращити комунікацію з відвідувачами. Автоматизувати рутинні процеси роботи, які виконують мед працівники	Сектор приватних медичних клінік	Автоматизація рутинних робочих процесів, ведення аналітичної статистичної інформації, з медичних установ різного типу, можливість інтегрувати різні медичні установи
	Державний сектор	
	Науковий сектор	

Електронне направлення надає можливість довільно обрати медзаклад. Навіть якщо людина зареєстрована в одній лікарні, вона зможе потрапити в іншу незалежно від прописки.

В даній системі є багато переваг, велика частина приватних медичних закладі підключена до систем медичної інформаційної інфраструктури. Та всі державні медичні заклади співпрацюють з нею.

Можливості запуску реалізації ідеї на ринку.

На даному му етапі ринок інформаційних технологій в Україні дуже стрімко розвивається. Існує велика кількість іт компаній які працюють в Україні. Розробка медичної інформаційної інфраструктури відноситься до інформаційних технологій. Проте варто зазначити, що попит на ринку інформаційних технологій також великий.

Також варто зазначити, що все ж таки розробка медичної інформаційної інфраструктури в Україні посідає певну нішу, і наразі держава вже створила медичну інформаційну систему.

Проте залишається велика ніша в приватному секторі розробки медичної інформаційної інфраструктури. Характеристика ринку таблиця 5.2.

Таблиця 5.2 – Попередня характеристика потенційного ринку

№	Показники стану ринку (найменування)	Характеристика
1	Головні конкуренти	Локальні розробники медичної інформаційної інфраструктури, та загально державні.
2	Динаміка ринку (якісна оцінка)	Зростає
3	Наявність обмежень для входу (вказати характер обмежень)	Великі фінансові інвестиції, сертифікація.
4	Специфічні вимоги до стандартизації та сертифікації	Вимагає розробки системи захисту даних, яка має сертифікуватися державою.

Наразі в Україні головним чином, напрямок розвитку медичної інформаційної інфраструктури визначає держава. Саме державні клінік в Україні з 2016 року поетапно, повністю всі підключилися до єдиної медичної інформаційної інфраструктури.

Звісно важливу роль грають також користувачі, які в першу чергу хочуть записуватися без черг. Варто зазначити, що державна медична інформаційна інфраструктура передбачає також підключення приватних установ. Також варто відмітити, що приватні медичні заклад зараз також активно впроваджують медичну інформаційну інфраструктуру.

Попередня характеристика потенційних клієнтів таблиця 5.3.

Таблиця 5.3 – Попередня характеристика потенційних клієнтів

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів (користувачів)
1	Державне замовлення на створення медичної інформаційної інфраструктури	Державні заклади	Системи повинні бути великими, та високонавантаженими.	Зрозуміла система керування, чітка робота системи, надійність
2	Приватні замовники розробки медичних інформаційних систем	Приватні заклади	Системи можуть буди невеликих обсягів, не потребують держсертифікації	Системні вимоги розробляє конкретний приватний заклад

При розробці проектів медичної інформаційної інфраструктури треба провести чіткий аналіз спроможностей даної системи, спроможностей фірми яка її розробляє. зрозуміти, можливі труднощі, та виробити план дій по їх подоланню.

Варто також проаналізувати економічну обстановку в державі, виявити економічні можливості. Зрозуміти економічні ризики. Побудувати відповідну економічну модель діяльності компанії.

5.2 Розробка ринкової стратегії

Вагомою частиною плану є просування виготовленого продукту на ринку. Для цього треба проаналізувати ринки. Аналіз показан в таблиці 5.4.

Таблиця 4 – SWOT-аналіз

Сильні сторони:	Слабкі сторони:
<ul style="list-style-type: none"> – затребуваність на ринку; – висока технологічність; – можливість інтеграції з іншими системами; – великий кадровий потенціал; – модульність системи 	<ul style="list-style-type: none"> – проект потребує великих інвестицій; – велика конкуренція у державному сегменті; – можливість текучки кадрів
Можливості:	Загрози:
<ul style="list-style-type: none"> – масштабування на загальнодержавний рівень; – участь у інвестиційних проектах як державних так і приватних; – співпраця з навчальними закладами-створення власних центрів підготовки розробників, створення власних технічних центрів для обслуговування обладнання, яке використовується в медичній інформаційній інфраструктурі. 	<ul style="list-style-type: none"> – нестабільність ринку; – фінансові коливання які можуть вплинути на роботу фірми, якщо вона отримує державне замовлення; – соціальні чинники, та обмеження які вводяться державою по переміщенню осіб, та їх можливості знаходитися на робочих місцях.

Можливі ринкові стратегії.

Використання стратегії диференційованого маркетингу, реалізована як маркетингова програма, адаптована під конкретний ринковий сегмент. Ця стратегія дозволяє їй компанії діяти у різних сегментах з індивідуальною, ціновою, продажною та комунікаційною стратегією. Ціни продажів встановлюються з урахуванням цінової чутливості кожного сегмента. Подібна стратегія передбачає значні витрати на кожний сегмент і націлена на великий ринок, пропонуючи безліч індивідуалізованих видів товарів, що можуть сильно відрізняються один від одного, призначених для задоволення численних ринкових сегментів.

Стратегія недиференційованого маркетингу, що полягає у тому, що всі сегменти ринку сприймаються однаково. Суть цієї стратегії стандартизації полягає в економії на виробничих витратах, просуванні.

Концентрована стратегія це концентрує свої зусилля одному чи кількох основних сегментах ринку, розробляє маркетингові підходи і продукцію спеціально для задоволення потреб саме цих груп покупців.

Відповідно до цієї стратегії товар зобов'язаний максимально відповідати потребам відповідного сегменту ринку. По кожному сегменту ринку компанія виробляє окрему маркетингову програму, варто зазначити, що це пов'язано з побудовою довгострокових стратегічних цілей та збільшенням витрат.

Стратегія розробки нового товару – ведеться пошук нових товарів, які в технологічному і ринковому відношенні були б схожі вже виробленим товарам підприємства і залучали б нових покупців.

Стратегія продовження товару – новий товар є продовженням попередніх товарів вже випускається, розрахований на коло покупців, що сформувалося, його виробництво ведеться без серйозних змін прийнятої на підприємстві технології.

Стратегія конгломератної диверсифікації – розпочинається випуск нового товару, не пов'язаного з виробленими досі підприємствами, тому

потрібна розробка нових технологій та освоєння нових ринків. Це найбільш трудомістка стратегія, яка потребує значних ресурсних витрат.

Сегментно орієнтована стратегія, чи стратегія ринкової ніші, заснована на низьких витратах на виробництво та розробку, орієнтована на певний сегмент сегмент покупців, де підприємство випереджає своїх конкурентів з допомогою нижчих витрат виробництва.

Сфокусована стратегія, заснована на диференціації продукції, ставить за мету забезпечення споживачів обраного сегмента товарами чи послугами, що найбільш повно відповідають їх потребам цільові групи таблиця 5.5.

Також є основні принципи побудови подібних стратегій :

- постійний розвиток, оновлення продукції;
- організовувати побудову служби маркетингу;
- складання бюджету та плану маркетингу в цілому;
- маркетинговий контроль.

Таблиця 5.5 – Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Державні заклади	Середня	Середній	Висока	Висока
2	Приватні заклади	Висока	Високий	Середня	Висока
3	Університети	Середня	Середній	Середня	Середня
Які цільові групи обрано: Державні заклади, приватні медичні заклади, університети					

5.3 Склад, чисельність та фонд заробітної плати виробничих працівників

Всі нарахування та утримання працівників організацій як правило можна на такі групи.

Періодичні (планові). Про необхідність їх нарахування (утримання) відомо заздалегідь, їхній розрахунок виконується регулярно щомісяця. Для них не потрібно введення додаткової інформації на конкретний місяць;

Разові. Їх нарахування (утримання) провадиться у разі потреби, при цьому потрібне введення додаткової інформації.

Разові нарахування та утримання.

До разових нарахувань/утримань можна віднести: оплата невідпрацьованого часу (наприклад, відпускні, лікарняні); премії та інші разові доплати; штрафи та інші разові утримання.

Згідно з чинним законодавством України як було закладено у Законі

«Про державний бюджет на 2021 рік» від 15.12.2020 року № 1082. Мінімальна заробітна плата на грудень 2021 року в Україні становить 6500 гривень. Відповідно погодинна оплата розраховується за грудень 2021 так: робочих годин на рік однаково 1994 (40-годинний робочий тиждень). Тоді погодинна МЗП = $6500 \times 12 / 1994 \approx 39,12$ грн.

В таблиці 5.6 наведено оплату праці працівників.

Таблиця 5.6 – Оплата праці

Категорії працівників	Наявна чисельність, осіб		Тарифна ставка за розрядом виконуваних робіт, грн.	Ефективний фонд робочого часу, годин	Тарифний заробіток, грн.	Преміальний відсоток до тарифного заробітку	Розмір премії, грн.	Річний фонд заробітної плати, грн.	ЄСВ, грн.
	за зміну	на добу							
1	2	3	4	5	6	7	8	9	10
Виробничі працівники, в тому числі:	2	2	70	8	11760	20%	2352	169344	37255,68
1. Основні працівники									
2. Допоміжні працівники									
3. Черговий та ремонтний персонал									
Разом виробничих працівників	1	2	X	X	X	X	2352	169344	37255,68

Згідно Закону України Про збір та облік єдиного внеску на загальнообов'язкове державне соціальне страхування, єдиний соціальний внесок становить 22 %.

Згідно штатного розпису на 2021 рік Національного університету «Запорізька політехніка» визначено наступні оклади професорсько-викладацькому складу (табл. 5.7, 5.8).

Таблиця 5.7 – Посадові оклади

Посада	Оклад по етс
Зав. каф. (к.н., доц.)	8150,00
Старший викладач (к.н., доц., засл. діяч)	8010,00
Зав. каф. (д.н., доц.)	9131,00
Зав. каф. (д.н., проф.)	9719,00
Викладач, асистент	7449,00

Таблиця 5.8 – Склад, чисельність та фонд заробітної плати адмінперсоналу

Посада	Кількість осіб	Посадовий оклад, грн	Преміальний відсоток до окладу, %	Сума премії, грн	Місячна заробітна плата, грн	Річний фонд оплати праці, грн	ЄСВ, грн
1	2	3	4	5	6	7	8
Зав.каф. (д.н.,доц.)	1	9131,00	10	914	10045	120540	26518.8
Разом управлінсько-гоперсоналу	1		X				26518.8

5.4 Розрахунок матеріальних витрат

Програмне забезпечення це нематеріальний актив. Згідно з наказом Про затвердження Національного положення (стандарту) бухгалтерського обліку визначає поняття нематеріального активу. Нематеріальний актив – немонетарний актив, який не має матеріальної форми та може бути ідентифікований. Проте для розробки програмного забезпечення повинні бути використані:

- спеціальна комп’ютерна техніка;
- комплектуючі до комп’ютерної техніки;
- інше програмне забезпечення;
- принтер;
- офісне приладдя.

Розрахунок матеріальних витрат наведено в таблиці 5.8.

Таблиця 5.8 – Розрахунок матеріальних витрат

Матеріальні витрати	Норматив у розрахунку на один. продук. (послуг)	Виробнича програма	Обсяг сировини	Ціна	Сума грн
1	2	3	4	5	6
Монітор для комп'ютера	1		1	3000	3000
Системний блок	1		1	7000	7000
Клавіатура	1		1	1000	1000
Миш	1		1	500	500
Додаткова відео карта	1		1	3000	3000
Принтер				3000	3000
Інтернет				100	100
Разом				17600	17600

5.5 Розрахунок вартості спожитих послуг

Згідно Положення про Національну комісію, що здійснює державне регулювання у сферах енергетики та комунальних послуг, затвердженого Указом Президента України від 10.09.2014 № 715/2014, НКРЕКП здійснює державне.

В межах наданих законодавством повноважень НКРЕКП встановлює порядки (методики) формування, розрахунку та встановлення державних регульованих цін і тарифів для суб'єктів природних монополій у сферах енергетики та комунальних послуг, а також для інших суб'єктів господарювання, що провадять діяльність на ринках у сферах енергетики та комунальних послуг, якщо відповідні повноваження надані Регулятору законом, у тому числі Порядок встановлення (формування) тарифів на

послуги з розподілу електричної енергії (затверджений Постановою НКРЕКП від 05.10.2018 №1175). Ціна наелектричну енергію становить відповідно до постанови КМУ від 28.12.2020 №1325 1.68 грн/кВт год. Тарифи на централізоване водопостачання та водовідведення формувались та встановлювались відповідно до Порядку формування тарифів на централізоване водопостачання та водовідведення, затвердженого постановою Національної комісії, що здійснює державне регулювання у сферах енергетики та комунальних послуг, від 10.03.2016 № 302, зареєстрованою у Міністерстві юстиції України 19.04.2016 за № 593/28723, які передбачають встановлення окремих тарифів для споживачів, які є суб'єктами господарювання у сфері централізованого водопостачання та водовідведення (ЦВВ), та Процедури встановлення тарифів на централізоване водопостачання та водовідведення, затвердженої постановою НКРЕКП від 24.03.2016 № 364, зареєстрованою у Міністерстві юстиції України 27.04.2016 за № 643/28773, на підставі документів та розрахункових матеріалів, наданих суб'єктами природних монополій у сфері централізованого водопостачання та водовідведення.[12]. Дані тарифи встановлені суб'єктам природних монополій у сфері централізованого водопостачання та водовідведення, ліцензування діяльності яких здійснюється НКРЕКП. Згідно постанови Постанова НКРЕКП від 16.12.2020 №2499 тарифи КП «Водоканал» на холодне водопостачання та водовідведення становлять 24,168 грн./м³ з ПДВ. Згідно рішення МВК та Постанова НКРЕКП від 30.11.2020 № 2253 (зміни до постанови від 14.01.2020 № 85) постачання гарячої води коштує 87,76 грн./м³, послуга з постачання теплової енергії коштує 1 655,08 грн./Гка.

Розрахунок вартості спожитих послуг таблиця 5.9.

Таблиця 5.9 – Розрахунок вартості спожитих послуг

Вид послуг	Норматив у розрахунку на один. продук. (послуг)	Виробнича програма	Обсяг послуг	Тарифи	Сума грн
1	2	3	4	5	6
Електропостачання	-	-	500кВ	1.68 грн/кВт год	840
Водопостачання	-	-	3 м3	15,312 грн\м3	45,936
Теплопостачання	-	-	1.5Гкал	1655,08 грн./Гкал	2482,623
Разом	-	-			3368,556

5.6 Розрахунок амортизації

Розрахунок амортизації наведено в таблиці 5.10.

Таблиця 5.10 – Розрахунок амортизації

Група основних засобів	Норма амортизації	Первісна вартість ОЗ на 01.01 2021 грн	Сума грн
1	2	3	8
Монітор для ком`пютера	50%	3000	1500
Системний блок	50%	7000	3500
Клавіатура	50%	1000	500
Миш	50%	500	250
Принтер	50%	3000	5750
Разом			7250

5.7 Кошторис витрат

Кошторис витрат наведено в таблиці 5.11.

Таблиця 5.11 – Кошторис витрат

Калькуляційні статті	Витрати	
	у розрахунку на одиницю продукції, грн.	у розрахунку на весь обсяг продукції, грн.
Сировина та матеріали	17600	17600
Електропостачання	840	840
Водопостачання	45,936	45,936
Теплопостачання	2482,623	2482,623
Інші		
Разом	21168,599	17868,559
Заробітна плата основних виробничих працівників	42336	169344
ЄСВ	9312	37255,68
амортизація	7250	7250
Витрати на утримання та експлуатацію основних засобів поточний ремонт	-	-
Загальновиробничі витрати	-	-
Виробнича собівартість	-	-
Адміністративні витрати	-	-
Інші витрати	-	-
Повна собівартість	80067	80067

Розрахунок фінансового результату наведено в таблиці 5.12.

Таблиця 5.12 – Розрахунок фінансового результату

№ Периоду	Обсяги виробництва (послуг)	Обсяги реалізації	Залишок нереалізованої продукції	Ціна	Дохід	Витрати	Фінансовий результат
	1 система	1 система	-	250000	250000	80067	169933
Разом							169933

Данна система розробляється під конкретний заклад, або університету. Відповідно ціна за систему, або способи оплати за користування формуються з кожним покупцем окремо.

5.8 Карта стейкхолдерів

Карта стейкхолдерів зображена на рисунку 5.8.

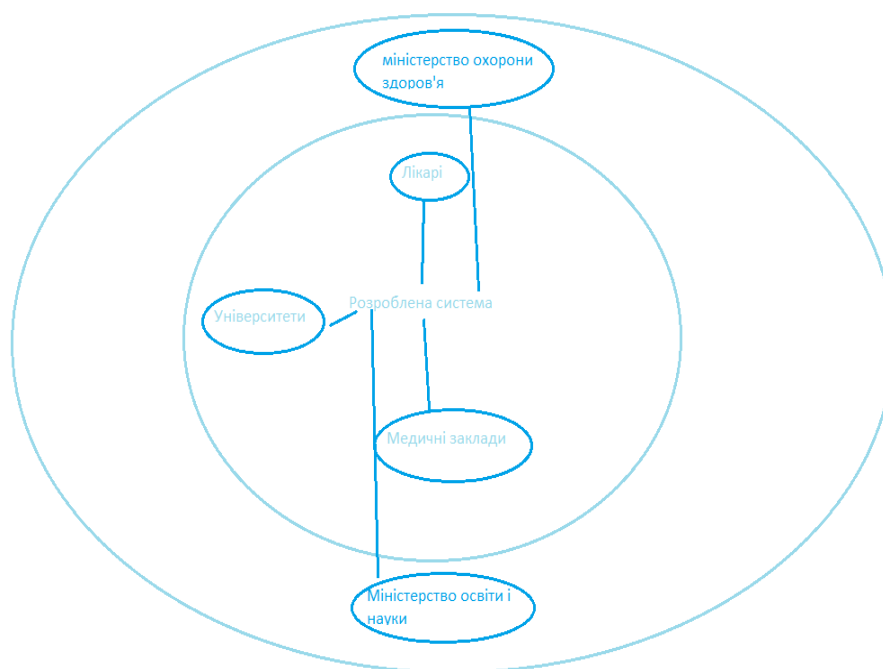


Рисунок 4.1 – Карта стейкхолдерів

5.9 Висновки

При розрахунку економічної частини даного проекту були виявлені основні економічні пункти які потребують великих фінансових витрат це в першу чергу заробітна платня.

Даний проект потребує масштабування, реальну його прибутковість можна визначити лише на практичному застосуванні Як показує практика подібних проекті прибутковість проекту залежить також від моделі фінансової взаємодії розробника та користувача.

З економічної точки зору такі проекти доцільно інтегрувати з іншими проектами даної галузі.

Даний проект можна також розвивати як дослідний по багатьом науково технічним напрямку, що в свою чергу опосередковано буде впливати на економічні показники. Варто зазначити, що при державній підтримці данини проект можна використовувати як платформу для навчання студентів технічним наукам, тобто йдеться не лише про прибутковість, а й про підвищення якості освіти.

6 ОХОРОНА ПРАЦІ І БЕЗПЕКА У НАДЗВИЧАЙНИХ СИТУАЦІЯХ

6.1 Аналіз потенційних небезпек

При розробленні диплому було виконано проектування медичної інформаційної інфраструктури приватної медичної установи. Специфіка роботи полягає в тому, що розробник постійно контактує з електронно – обчислювальними машинами, відповідно треба розглянути заходи безпеки при роботі з даним обладнанням. Робота ведеться у певному приміщенні, де також треба визначити заходи безпеки.

Основні небезпечні фактори:

- вплив електромагнітних полів від монітору комп'ютера. Може призвести до підвищеного стомлення, зниження працездатності та профзахворювання;
- недотримання санітарних норм внаслідок чого може виникнути погане самопочуття;
- недотримання параметрів мікроклімату знижує працездатність, погіршує самопочуття, може призвести до профзахворювань; непоінформованість персоналу у відповідній послідовності дій в умовах надзвичайної ситуації призводить до травмування та до інших непередбачених негативних наслідків;
- враження електричним струмом, може виникнути через недотримання вимог при прокладанні провідки, неправильній експлуатації обладнання;
- неправильне освітлення може бути причиною таких захворювань, як короткозорість, спазм, акомодация, зорове стомлення та інших хвороб, знижує розумову і фізичну працездатність; збільшує число помилок у виробничих процесах, аварій та нещасних випадків;

- невідповідність ергономічних характеристик робочого місця призводить до, зниження продуктивності робочого процесу та швидкого втомлення працівника;
- недотримання температурного режиму у приміщенні, в поєднанні з підвищеною вологістю та швидкістю повітря, впливає на посилення процесів тепловіддачі, що призводить до охолодження організму дитини і, як наслідок, – розвитку захворювань;
- недотримання рівня шуму у приміщенні призводить до перенавантаження слуху людини, та зниження ефективності праці
- недотримання норм розпорядку робочого дня приводить до розвитку втоми та перевтоми веде до порушення координації рухів, зорових розладів, неуважності, втрати пильності та контролю реальної ситуації; при цьому працівник порушує вимоги технологічних інструкцій, припускається помилок та неузгодженості в роботі; у нього знижується відчуття небезпеки крім того, перевтома супроводжується хронічною гіпоксією (кисневою недостатністю), порушенням нервової діяльності;
- шкідливе випромінювання електромагнітних полів або іонізуючих випромінювань, які мають напруженість, що перевищує гранично допустимий рівень, можуть викликати порушення з боку нервової, серцево-судинної систем, органів травлення і деяких біохімічних показників крові; таким чином, техногенні електромагнітні поля являються переважно шкідливими для людини; особливо велика шкода виникає в тому разі, коли рівні цих електромагнітних перевищують гранично допустимий рівень, встановлені санітарними нормами;
- невідповідність персоналу в умовах надзвичайних ситуацій, призводить до дезорганізації робочий колектив, не дозволяє вчасно зреагувати на виникнення позаштатних ситуацій, викликає велику паніку нерозуміння, ситуації.

6.2 Заходи по забезпечення техніки безпеки

Так як основну роботу працівник виконує за комп'ютером тому правила безпечної роботи, тобто техніка безпеки регламентується ДСанПіН 3.3.2.007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електроннообчислювальних машин» [24].

Відповідно до них розробляються правила. Встановлюються такі внутрішньозмінні режими праці та відпочинку при роботі з ЕОМ при 8-годинній денній робочій зміні в залежності від характеру праці:

- для розробників програм із застосуванням ЕОМ, слід призначати регламентовану перерву для відпочинку тривалістю 15 хвилин через кожен годину роботи за ВДТ;
- для операторів із застосування ЕОМ слід призначати регламентовані перерви для відпочинку тривалістю 15 хвилин через кожні дві години;
- для операторів комп'ютерного набору слід призначати регламентовані перерви для відпочинку тривалістю 10 хвилин після кожною години роботи за ВДТ.

При 12-годинній робочій зміні регламентовані перерви повинні встановлюватися в перші 8 годин роботи аналогічно перервам при 8-годинній робочій зміні, а протягом останніх 4-х годин роботи, незалежно від характеру трудової діяльності, через кожен годину тривалістю 15 хвилин.

Розміщення робочих місць з ВДТ ЕОМ і ПЕОМ у підвальних приміщеннях, на цокольних поверхах заборонено.

Площа на одне робоче місце має становити не менше ніж 6,0 м², а об'єм не менше ніж 20,0 м³.

Приміщення для роботи з ВДТ повинні мати природне та штучне освітлення відповідно до СНиП II-4-79.

Природне освітлення має здійснюватись через світлові прорізи, орієнтовані переважно на північ чи північний схід і забезпечувати коефіцієнт природною

освітленості (КПО) не нижче ніж 1,5%. Розраховується КПО за методикою, викладеною в СНиП II-4-79.

Виробничі приміщення для роботи з ВДТ (операторські, диспетчерські) не повинні межувати з приміщеннями, в яких рівні шуму і вібрації перевищують допустимі значення (виробничі цехи, майстерні тощо) за СН 3223-85, СН 3044-84, ГР 2411-81, ГОСТ 12.1.003-83.

Звукоізоляція огорожувальних конструкцій приміщень з ВДТ має забезпечувати параметри шуму, що відповідають вимогам СН 3223-85, ГОСТ 12.1.003-83 [25], ГОСТ 12.1.012-90.

Приміщення для роботи з ВДТ мають [25] бути обладнані системами опалення, кондиціонування повітря, або припливно-витяжною вентиляцією відповідно до СНиП 2.04.05-91. Нормовані параметри мікроклімату, іонного складу повітря [25], вмісту шкідливих речовин мають відповідати вимогам СН 4088-86, СН 2152-80, ГОСТ 12.1.005-88, ГОСТ 12.1.007-76.

Віконні прорізи приміщень для роботи з ВДТ мають бути обладнані регульованими пристроями (жалюзі, завіски, зовнішні козирки).

Для внутрішнього оздоблення приміщень з ВДТ слід використовувати дифузно-відбивні матеріали з коефіцієнтами відбиття для стелі 0,7-0,8, для стін 0,5-0,6.

Покриття підлоги повинне бути матовим з коефіцієнтом відбиття 0,3-0,5. Поверхня підлоги має бути рівною, неслизькою, з антистатичними властивостями. Забороняється для оздоблення інтер'єру приміщень ВДТ застосовувати полімерні матеріали (деревинно-стружкові плити, шпалери, що миються, рулонні синтетичні матеріали, шаруватий паперовий пластик тощо), що виділяють у повітря шкідливі хімічні речовини.

Виробничі приміщення можуть обладнуватись шафами для зберігання документів, магнітних дисків, полицями, стелажми, тумбами тощо з урахуванням вимог до площі приміщень. У приміщеннях з ВДТ слід щоденно робити вологе прибирання.

Приміщення з ВДТ мають бути оснащені аптечками першої медичної допомоги.

При приміщеннях з ВДТ мають бути обладнані побутові приміщення для відпочинку під час роботи, кімната психологічного розвантаження. В кімнаті психологічного розвантаження слід передбачити встановлення пристроїв для приготування й роздачі тонізуючих напоїв, а також місця для занять фізичною культурою (СНиП 2.09.04.-87).

Враховуючи, що електроживлення електрообладнання приміщення з ПК здійснюється від мережі змінного струму напругою 220 В і частотою 50 Гц. Відповідно до вимог «ПУЕ», електрообладнання в приміщенні з ПК, характеризується як електроустановки до 1000 В, тому згідно вимог глави 1.7 «ПУЕ», величина опору контуру захисного заземлення електрообладнання приміщення з ПК у будь-яку пору року не перевищує – 4 Ом.

6.3 Розрахунок захисного заземлення

Захисне заземлення – це навмисне електричне з'єднання з землею або її еквівалентом металевих неструмоведучих частин, які можуть виявитися під напругою [27] (табл. 6.1).

Таблиця 6.1 – Вихідні данні для розрахунку захисного заземлення

Назва параметру	Одиниці вимірювання	Величина параметру
Наруга	U, В	380
Розміри приміщення	L×B, м	8X4
Опір природного заземлювача	Re, Ом	12
Довжина вертикального стрижня	l_B , м	5
Діаметр вертикального стрижня	d, м	0.01

Продовження таблиця 6.1

Назва параметру	Одиниці вимірювання	Величина параметру
Ширина смуги	$b_c, \text{м}$	0.028
Питомий опір ґрунту	$\rho_{\text{вим}}, \text{Ом} \cdot \text{м}$	84
Коефіцієнт сезонності ґрунту	Ψ	1.1
Розрахунковий опір заземлення, при $U < 1000 \text{ В}$	$R_3, \text{Ом}$	4

Визначаємо необхідний опір штучного заземлювача R_u , Ом, якщо передбачається використання також природного заземлювача R_e , за формулою:

$$R_u = \frac{R_e R_3}{R_e - R_3} \text{ Ом}, \quad (6.1)$$

де R_e – опір розтіканню струму природних заземлювачів, Ом;

R_3 – розрахунковий нормований опір заземлювального пристрою (ЗП),
 $R_e = 12$, $R_3 = 4$.

$$R_u = \frac{R_e R_3}{R_e - R_3} = \frac{12 \cdot 4}{12 - 4} = 6 \text{ Ом}. \quad (6.2)$$

Визначимо розрахунковий питомий опір ґрунту за формулою:

$$\rho_{\text{в(Г)}} = \rho_{\text{вим}} \Psi_{\text{в(Г)}}, \quad (6.3)$$

де $\rho_{\text{в(Г)}}$ – розрахунковий питомий опір ґрунту, Ом·м;

$\rho_{\text{вим}}$ – питомий опір землі, отриманий у результаті вимірів, Ом·м;

$\rho_{\text{вим}} = 84 \text{ Ом} \cdot \text{м}$;

$\Psi_{в(г)}$ – коефіцієнт сезонності, що враховує промерзання чи висихання ґрунту; $\Psi_{в(г)} = 1.1$.

$$\rho_{в(г)} = \rho_{вим} \Psi_{в(г)} = 84 * 1.1 = 92.4$$

Обчислимо опір розтіканню струму одиночного вертикального заземлювача R_B , Ом. Для стрижневого заземлювача круглого перерізу, заглибленого в ґрунт (рисунок 6.1), розрахункова формула має вигляд:

$$R_B = \frac{\rho_B}{2 * \pi * l_B} \left(\ln \frac{2 * l_B}{d} + \frac{1}{2} * \ln \frac{4 * t_B + l_B}{5 * t_B - l_B} \right) = 10, \text{ Ом} \quad (6.4)$$

де ρ_B – розрахунковий питомий опір ґрунту, Ом·м; $\rho = 92.4$ Ом·м;

l_B – довжина вертикального стрижня 5 м;

d – діаметр перерізу стрижня, м; $d = 0,01$ м;

t_B – відстань від поверхні ґрунту до середини довжини вертикального стрижня, яка обчислюється за формулою:

$$t_B = 0.8 + \frac{1}{2} * l_B \quad (6.5)$$

$$t_B = 0.8 + \frac{1}{2} * l_B = 0.8 + 0.5 * 5 = 3.3 \text{ м.}$$

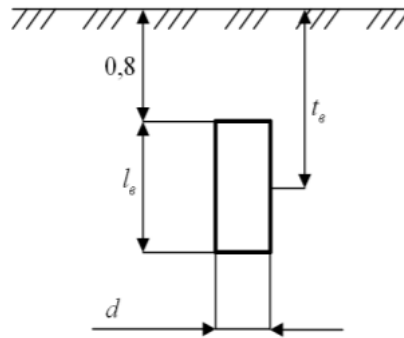


Рисунок 6.1 – Схема залягання стрижневого трубчастого заземлювача

Розрахуємо наближену (мінімальну) кількість вертикальних стрижнів:

$$n' = \frac{R_B}{R_U}, \text{шт} \quad (6.6)$$

де R_B – опір розтікання струму одиночного вертикального заземлювача, Ом;

$R_B = 10$ Ом;

R_U – необхідний опір штучного заземлювача, Ом; $R_U = 6$ Ом.

$$n' = \frac{R_B}{R_U} = \frac{10}{6} = 1.7 \text{ шт.}$$

Якщо виходити з розмірів контуру:

$$n = \frac{P}{a} = 4.8 \text{ шт. (загалом 5 штук),}$$

де периметр прямокутника (приміщення) який визначається формулою:

$$P = 2 * (L + B), \text{ м.}$$

$$P = 2 * (L + B) = 2 * 12 = 24 \text{ м} \quad (6.7)$$

$$a = k * l_B = 1 * 5 = 5 \quad (6.8)$$

де l_B – довжина вертикального стрижня, м;

k – коефіцієнт кратності, який дорівнює 1, 2, 3 (для заглиблених стаціонарних заземлювачів $k=1$);

a – відстань між стрижнями обумовлена зі співвідношенням.

Отриману кількість стрижнів округляють до більшого довідкового значення, але не менше n .

Визначаємо довжину горизонтальної смуги при конфігурації групового заземлювача – контур:

$$l_r = 1.05 * a * n \quad (6.9)$$

$$l_r = 1.05 * a * n = 5 * 5 = 26.25 \text{ м},$$

де n – кількість вертикальних стрижнів;

a – відстань між стрижнями, м.

Обчисліть опір розтіканню струму горизонтальної з'єднуючої смуги Ом. У випадку горизонтального смугового заземлювача рисунок1 розрахунок виконується за формулою:

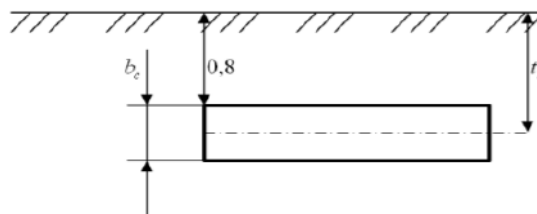


Рисунок 6.2 – Схема горизонтального смугового заземлювача

$$R_r = \frac{\rho_B}{2 * \pi * l_r} \left(\ln \frac{2 * l_r^2}{b_c * t_r} \right) = 6.2 \text{ Ом}, \quad (6.10)$$

де $\rho_{\text{в}}$ – розрахунковий питомий опір ґрунту 92.4 Ом* м;

$l_{\text{Г}}$ – довжина горизонтальної смуги 26.25 м;

$b_{\text{с}}$ – ширина смуги 0.028 м.

Відстань від поверхні ґрунту до середини и горизонтальної смуги ,яка обчислюється за формулою:

$$t_{\text{Г}} = 0.8 + \frac{1}{2} * b_{\text{с}}, \text{ м} \quad (6.11)$$

$$t_{\text{Г}} = 0.8 + \frac{1}{2} * b_{\text{с}} = 0.8 + 0.5 * 0.028 = 0.81 \text{ м.}$$

Розрахуємо еквівалентний опір розтіканню струму групового заземлювача.

$$R_{\text{гр}} = \frac{R_{\text{в}} R_{\text{Г}}}{R_{\text{в}} \eta_{\text{Г}} + R_{\text{Г}} \eta_{\text{в}} n} = 2.3 \text{ Ом}, \quad (6.12)$$

де $R_{\text{в}}$ – опір розтікання струму одиничного вертикального заземлювача 10 Ом;

$R_{\text{Г}}$ – опір розтікання струму горизонтальної смуги 6.2 Ом;

$\eta_{\text{Г}}, \eta_{\text{в}}$ – коефіцієнти використання вертикальних стрижнів і горизонтальної смуги $\eta_{\text{в}} = 0.69, \eta_{\text{Г}} = 0.45$;

n – кількість вертикальних стрижнів 5 штук.

Отриманий опір розтіканню струму групового заземлювача не повинен перевищувати необхідний опір визначений у пункті 2.

$$R_{\text{гр}} < R_{\text{н}},$$

$$2.3 < 4 \text{ Ом.}$$

6.4 Заходи безпеки у надзвичайних ситуаціях

6.4.1 Заходи з пожежної безпеки

Конструкторське бюро згідно з ДСТУ Б В.1.1-36:2016 «Визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою» відноситься до категорії «П-Па», а клас можливої пожежі, згідно ДСТУ EN 2:2014 «Класифікація пожеж (EN 2:1992, EN 2:1992/A1:2004, IDT)», визначається як «А».

Оскільки приміщення належить до категорії «А» з пожежної небезпеки, тому відповідно до вимог ДБН В.1.1-7:2016 «Пожежна безпека об'єктів будівництва. Загальні вимоги» воно має I ступінь вогнестійкості.

У разі виникнення пожежі у приміщенні конструкторського бюро, обладнаному ПК з ВДТ для евакуації персоналу відповідно до вимог ДБН В.1.1-7:2016 «Пожежна безпека об'єктів будівництва. Загальні вимоги» передбачені виходи, по обидві сторони приміщення, з одного боку вікно (на пожежні сходи), а з іншого – вхідні двері. Згідно п. 2.29 (табл. 2) СНиП 2.09.02-85* «Производственные здания», відстань від найбільш віддаленого робочого місця до найближчого евакуаційного виходу не обмежується.

Обладнання, силові та освітленні мережі приміщення конструкторського бюро обладнаного ПК з ВДТ відповідають вимогам пожежної безпеки, оскільки виконані відповідно до вимог НПАОП 40.1-1.32-01 «Правила будови електроустановок. Електрообладнання спеціальних установок», та мають ступінь захисту ізоляції обладнання IP44 яка відповідає класу пожежанебезпечної зони П-Па до якої належить приміщення.

Згідно вимог ДБН В.2.5-56:2014 «Системи протипожежного захисту», в приміщенні конструкторського бюро, обладнаному ПК з ВДТ встановлена система пожежної й охоронної сигналізації «Сигнал-ВК6». Яка забезпечує виявлення теплових і димових ознак пожежі і місця виникнення пожежі з точністю до місця розміщення датчика.

Згідно з НАПБ А.01.001-14 «Правил пожежної безпеки в Україні» на кожному об'єкті відповідним документом (наказом, інструкцією тощо) повинен бути встановлений протипожежний режим, який включає:

- порядок утримання шляхів евакуації;
- визначення спеціальних місць для куріння;
- порядок застосування відкритого вогню;
- порядок використання побутових нагрівальних приладів;
- порядок проведення тимчасових пожежонебезпечних робіт;
- правила проїзду та стоянки транспортних засобів;
- місця для зберігання і допустиму кількість сировини, напівфабрикатів та готової продукції, що можуть одночасно знаходитися у приміщеннях і на території;
- порядок прибирання горючого пилу й відходів, зберігання промасленого спецодягу та ганчір'я, очищення елементів вентиляційних систем від горючих відкладень;
- порядок відключення від мережі електроживлення обладнання та вентиляційних систем у разі пожежі;
- порядок огляду й зачинення приміщень після закінчення роботи;
- порядок проходження посадовими особами навчання й перевірки знань з питань пожежної безпеки, а також проведення з працівниками протипожежних інструктажів та занять з пожежно-технічного мінімуму з призначенням відповідальних за їх проведення;
- порядок організації експлуатації і обслуговування наявних засобів протипожежного захисту;
- порядок проведення планово-попереджувальних ремонтів та оглядів електроустановок, опалювального, вентиляційного, технологічного та іншого інженерного обладнання;
- порядок збирання членів пожежно-рятувального підрозділу добровільної пожежної охорони та посадових осіб, відповідальних за

пожежну безпеку, у разі виникнення пожежі, виклику вночі, у вихідні й святкові дні;

– порядок дій у разі виникнення пожежі: порядок і способи оповіщення людей, виклику пожежно-рятувальних підрозділів, зупинки технологічного устаткування, вимкнення ліфтів, підйомників, вентиляційних установок, електроспоживачів, застосування засобів пожежогасіння; послідовність евакуації людей та матеріальних цінностей з урахуванням дотримання техніки безпеки.

6.4.2 Заходи з цивільного захисту

Індивідуальний спосіб захисту передбачає застосування засобів індивідуального захисту органів дихання, шкіри, а також медичних засобів захисту. Цей спосіб широко застосовують у мирний час в умовах радіоактивного забруднення.

В зонах, заражених сильнодіючими отруйними речовинами, осередках біологічного зараження, районах стихійних лих. У режимі надзвичайної ситуації і надзвичайного стану всі заходи, які передбачається застосовувати для захисту населення, включають застосування засобів індивідуального захисту (ЗІЗ). Засоби індивідуального захисту призначаються для захисту від потрапляння у середину організму людини, а також на шкіру та одяг отруйних речовин (ОР), сильнодіючих отруйних речовин (СДОР), радіоактивних речовин (РР) та бактеріальних (біологічних) засобів (БЗ) ураження.

Засобами захисту органів дихання є фільтруючі, ізолюючі протигази, респіратори, протипилові тканинні маски та ватно-марлеві пов'язки.

Фільтруючі протигази призначені для захисту органів дихання, обличчя та очей від радіоактивних, отруйних і сильнодіючих отруйних речовин, біологічних засобів та інших небезпечних речовин. Принцип захисної дії фільтруючого протигазу базується на очищенні (фільтрації)

повітря, що вдихується від шкідливих сумішей, у фільтропоглинаючій коробці. 9 Фільтруючі протигази є основними і найбільш поширеними засобами для захисту органів дихання.

Для захисту органів дихання від окису вуглецю (СВ) застосовується комплект додаткового патрона ДП-2 з лицьовою частиною протигазів ГП-5 і ГП-7, при цьому зміст кисню в повітрі повинен бути не менш 18%, також може бути використаний гопкалітовий патрон ДП-1.

Цивільний протигаз ГП-5 використовується з 60-х років минулого століття. Фільтропоглинаюча коробка (ФПК) має незначний опір диханню, невелику вагу і габарити, за своїми захисними властивостями практично не відрізняється від ФПК попередніх моделей, за винятком часу захисної дії. ФПК в робочому положенні приєднується безпосередньо до лицьової частини. 10 Протигаз ГП-5 комплектується лицьовою частиною ШМ-62 (ШМ62У), які випускаються 5-ти розмірів. Конструкція каналів надходження повітря в підмасочний простір дає змогу зменшити ефект запотівання окулярів, а застосування незапотіваючих плівок повністю його усуває. Модифікований протигаз ГП-5М комплектується лицьовою частиною ШМ-62МУ.

Респіратор Р-2 – являє собою фільтруючу напівмаску, споряджену одним видихальним клапаном із запобіжним екраном, двома вдихальними клапанами, оголовком, який складається з еластичних шворок, що не розстібуються, і носового затискача. Зберігається респіратор в поліетиленовому пакеті з кільцем.

Респіратори Р-2 виготовляються трьох розмірів. Розмір вказується на внутрішній підборідній частині напівмаски. Зовнішня частина напівмаски виготовлена з поліуретану (пористого синтетичного матеріалу), а внутрішня – з тонкої повітронепроникної плівки, в яку вставлені клапани для вдиху. Між поліуретаном і плівкою розміщений фільтр з полімерних волокон.

До найпростіших засобів захисту органів дихання Ватно-марлева пов'язка.

Засоби захисту шкіри призначаються для захисту шкіряних покривів, обмундирування, взуття і спорядження від зараження отруйними, радіоактивними речовинами і бактеріальними засобами, а також є тимчасовим захистом від запалювальних речовин, світлового випромінювання та інших небезпечних факторів. За призначенням засоби захисту шкіри поділяються на загальновійськові, спеціальні і підручні. До загальновійськових належать засоби, призначені для захисту особового складу: загальновійськовий захисний комплект (ЗЗК), загальновійськовий комплексний захисний костюм (ЗКЗК) і імпрегноване обмундирування. Спеціальний захисний одяг застосовується при роботі на зараженій місцевості, при роботі з ОР, СДОР. Такими засобами захисту є легкий захисний костюм Л-1 і захисні комплекти ЗК-1, ЗК-2, ЗК-3.

Загальновійськовий захисний комплект (ЗЗК) використовується для захисту від РР, ОР, СДОР і БЗ шкіряних покривів, обмундирування і спорядження. Він, як правило, використовується в комплексі з імпрегнованим обмундируванням. До складу ЗЗК (входять захисні плащ, панчохи, рукавички.

Захисний плащ виготовляється зі спеціальної прогумованої тканини. Конструкція захисного плаща дозволяє не тільки вдягати його в рукава або використовувати як накидку, але й у вигляді комбінезона, в залежності від обставин. Захисні плащі виготовляють п'яти розмірів. Захисні панчохи виготовляються з прогумованої тканини. Підошви посилені гумовою сполукою. Захисні панчохи кріпляться за допомогою хлястиків і шворок. Вони мають три розміри. Захисні рукавички – гумові, виготовляються двох видів: літні БЛ-1М та зимові БЗ-1М. Літні рукавички п'ятипалі, зимові – двопалі. Загальновійськові захисні комплекти, згорнуті в чохли або вкладені в сумки, зберігаються в кімнаті зберігання майна або в спеціальних шафах (стелажах).

Легкий захисний костюм Л-1 – виготовлений з прогумованої тканини і складається з куртки з капюшоном, брюк з панчохами, двопалих рукавичок

і підшоломника. В комплект входить сумка для перенесення костюма і запасна пара рукавичок. При зберіганні засобів захисту шкіри, виготовлених з гуми або прогумованої тканини, необхідно враховувати, що гума при зберіганні підлягає процесу старіння, в результаті чого на її поверхні з'являються тріщини. Тому, для зменшення процесу старіння, засоби захисту шкіри з гуми або прогумованої тканини не можна зберігати в світлих (незатемнених) приміщеннях, особливо в умовах проникнення в сховища прямих сонячних променів та постійної дії протягів. Вони зберігаються в сухих, неопалюваних сховищах упакованими в стандартні дерев'яні ящики.

ВИСНОВКИ

При виконанні магістерської роботи було проведено аналіз медичної інформаційної інфраструктури в Україні. Були виявлені основні напрямки її розвитку. Проведений функціональний аналіз даних систем виявлені їхні основні функції це такі:

- можливість обирати лікаря;
- можливість обрати мед заклад;
- робота з державними закладами;
- робота з приватними закладами;
- можливість обирати графік.

Проте виявлені їх недоліки цезокрема те, що вони не надають статистику яку міг би використовувати відвідувач.

Також данні системи мають низьку ступінь автономності , що може зменшити гнучкість системи.

Тому при розробці медичної інформаційної інфраструктури я врахував ці фактори та вніс їх в розроблювану систему.

У розробленій системі користувач при записі до лікаря може дивитися всі свої записи для цього на головній панелі була виведена спеціальна кнопка.

В системі передбачена можливість реєстрації відвідувачів. Також розроблений інтерфейс керування графіком лікаря.

Така ж функція була розроблена і для направлення тобто можна дивитися які препарати використовуються для конкретного направлення.

Розроблена медична інформаційна інфраструктура може використовуватися як функціональна система, причому вона має велику модульність і може змінювати властивості в залежності від потреб.

Логіка системи передбачає те, що вона буде складатися із багатьох автономних модулів, які будуть взаємодіяти з таблицями даних мета рівня, і кожна частина буде мати власні таблиці локального рівня, що значно

покращить роботу системи. За допомогою такого принципу робота впорядкується вразі.

Варто зазначити, що системи подібного типу можуть також використовуватися як для приватних лікарів, так і для державних установ.

Її перевагою є те, що вона може використовуватися як платформа для розробки більшої кількості інших систем.

Основною її перевагою є те, що на відміну від інших медичних інформаційних систем дозволяє показувати користувачу статистику препаратів використаних до діагнозів, та статистику діагнозу і використаних до нього препаратів.

Також великою перевагою даної системи, на відміну від інших інформаційних систем розглянутих вище є те, що кожний лікар в системі має індивідуальні таблиці для створення графіку та індивідуальні таблиці за допомогою яких роблять запис до цього лікаря.

Медична інформаційна інфраструктура є ваговою компонентною загальної система. Вона дозволяє покращити роботи закладів, збільшити продуктивність роботи конкретного працівника, зробити більш мобільним передачу медичних даних.

Для користувача системи надає можливість швидкої реєстрації, перегляд медичних даних. Зручні можливості для аналізу використання лікарських засобів.

Варто зазначити такі системи підвищують загальну комунікацію.

ПЕРЕЛІК ПОСИЛАНЬ

1. Анализ требований к информационным системам. [Электронный ресурс]. Режим доступа: <https://ivan-shamaev.ru/wp-content/uploads/2013/06/Information-systems-analysis-and-requirements-analysis.pdf>.
2. Распределенные вычислительные системы: учебное пособие [Электронный ресурс]. Режим доступа: http://window.edu.ru/catalog/pdf2txt/646/76646/57836?p_page=4
3. Различные архитектурные решения, используемые при реализации многопользовательских СУБД. [Электронный ресурс]. Режим доступа: <https://intuit.ru/studies/courses/508/364/lecture/8643?page=2>
4. Технология «Клиент – сервер». [Электронный ресурс]. Режим доступа: https://works.doklad.ru/view/b_WBz8swNHk.html
5. Многопользовательский режим работы с базой данных. Модели «клиент–сервер» в системах баз данных. Архитектура серверов баз данных. [Электронный ресурс]. Режим доступа: http://eos.ibi.spb.ru/umk/5_8/5/5_R0_T6.html#2_2
6. Технология модели клиент-сервер. [Электронный ресурс]. Режим доступа: <https://theslide.ru/uncategorized/tehnologiya-modeli-klient-server>
7. Основные типы клиент-серверной архитектуры. [Электронный ресурс]. Режим доступа: <http://csaa.ru/dva-osnovnyh-tipa-klient-servernoj-arhitektury/>
8. Модель технологии файл-сервер. [Электронный ресурс]. Режим доступа: https://studref.com/516445/ekonomika/bazovye_servernye_arhitektury
9. Архитектуры информационных систем. [Электронный ресурс]. Режим доступа: http://k504.xai.edu.ua/html/ucheba/rss/RSS_Lekciya_2.pdf
10. What is Python? Executive Summary. [Электронный ресурс]. Режим доступа: <https://www.python.org/doc/essays/blurb/>

11. Классы и объекты. ООП (объектно ориентированное программирование). [Электронный ресурс]. Режим доступа: <https://docplayer.com/62725784-Klassy-i-obekty-oop-obektno-orientirovannoe-programmirovanie-paradigma-programmirovaniya-v-kotoroy.html>
12. Структура HTML-документа. [Электронный ресурс]. Режим доступа: <http://www.codeharmony.ru/materials/33>
13. Html <!DOCTYPE> Описание. [Электронный ресурс]. Режим доступа: https://html5css.ru/tags/tag_doctype.php
14. Html стилі. [Электронный ресурс]. Режим доступа: <https://iit-web-lectures.readthedocs.io/ru/latest/www/css.html>
15. Устройство IDE.[Электронный ресурс]. Режим доступа: <https://www.internet-technologies.ru/articles/10-luchshih-ide.html>
16. Розробка веб -сайтів. Введення в каскадні таблиці стилів. [Электронный ресурс]. Режим доступа: http://blogs.kpi.kharkov.ua/v2/orobinska/wp-content/uploads/sites/30/2021/03/LK_4_Stili_CSS.pdf
17. Возможности PyCharm. [Электронный ресурс]. Режим доступа: <https://www.jetbrains.com/ru-ru/pycharm/features/>
18. Методика MTV (или MVC). [Электронный ресурс]. Режим доступа: <https://djbook.ru/ch05s02.html>
19. Методичні вказівки до лабораторних робіт з дисципліни “Медично інформаційна інфраструктура”. [Электронный ресурс]. Режим доступа: http://eir.zntu.edu.ua/bitstream/123456789/4909/1/Tabunshchyk_Methodological.pdf
20. Зв’язки в базах даних багато до багатьох. [Электронный ресурс]. Режим доступа: https://djbook.ru/rel1.9/topics/db/examples/many_to_many.html
21. Зв’язки в базах даних один до багатьох. [Электронный ресурс]. Режим доступа: https://djbook.ru/rel1.9/topics/db/examples/many_to_one.html

22. Зв'язки в базах даних один до одного. [Електронний есурс]. Режим доступу: https://djbook.ru/rel1.9/topics/db/examples/many_to_many.html
23. Как разворачивать с помощью WSGI. [Електронний есурс]. Режим доступу: <https://django.fun/docs/django/ru/3.2/howto/deployment/wsgi/>
24. ДСанПіН 3.3.2.007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електроннообчислювальних машин».
25. ДСН 3.3.6-042-99 «Санітарні норми мікроклімату виробничих приміщень».
26. ГОСТ 12.1.005-88 Загальні санітарно-гігієнічні вимоги до повітряної робочої зони.
27. ГОСТ 12.2.007.0-75* (2001) «ССБТ. Изделия электротехнические. Общие требования безопасности».

ДОДАТОК А

Текст програми

```
from django.http import request, HttpResponseRedirect, HttpResponseNotFound,
HttpResponse, response
from django.shortcuts import render, redirect
from django.template import RequestContext
from django.urls import reverse_lazy
from django.views.generic import CreateView, ListView

from.forms import *
from.models import *

from django.contrib.auth.views import LoginView

from django.contrib.auth import logout, login


def user_check_list(request,):
    context = {'Podstanovka': grafA.objects.filter(ures=request.user)}
    return render(request, 'DPU/v_VR.html', context )

def HTTT(request):

    return render(request, 'DPU/base.html')


class Registration_doctor(CreateView):
    form_class=grafForms
    template_name = 'DPU/register_doc.html'
    success_url = reverse_lazy('vvr')
```

```

def form_valid(self, form):
    dz=form.cleaned_data
    form.instance.ures = self.request.user

    if grafA.objects.filter(Month=dz['Month'], ChisA=dz['ChisA'],
grafa=dz['grafa']).count()==0:
        return super(Registration_doctor, self).form_valid(form)
    else:
        pass
        #return redirect("")

class Doctor_list_Month( CreateView):
    form_class=MonthForms
    template_name = 'DPU/Month.html'
    success_url = reverse_lazy('vvr')
    # permission_required='DPU.view_montha'
class Doctor_list_Day(CreateView):
    form_class = DayForms
    template_name = 'DPU/Day.html'
    success_url = reverse_lazy('vvr')

class Doctor_list_taim(CreateView):
    form_class = TimeForms
    template_name = 'DPU/Taim.html'
    success_url = reverse_lazy('dllt')

def delite_taim(request):
    CifriA.objects.all().delete()
    return redirect('vvr')
def delite_Month(request):

```

```

de=MonthA.objects.all().delete()
return redirect('vvr')
def delite_Day(request):
    de=ChisA.objects.all().delete()
    return redirect('vvr')

def delete_registration_user(request, id):
    try:
        person = grafA.objects.filter(id=id)
        person.delete()
        return redirect("vvr")
    except grafA.DoesNotExist:
        return HttpResponseRedirect("<h2>Person not found</h2>")

class Email_otpravka(CreateView):
    form_class=Pismo
    template_name = 'DPU/pochta.html'
    success_url = reverse_lazy('vvr')

    def form_valid(self, form):
        mail=send_mail(form.cleaned_data['name'],form.cleaned_data['name'],'dzusp@ra
mbler.ru,['dzusp@rambler.ru'],fail_silently=False)
        if mail:
            print('Отп')
        else:
            pass
        return redirect("/")

```

```

class uchet_stat(CreateView):
    form_class=statistica1Forms
    template_name = 'DPU/statictics.html'
    success_url = reverse_lazy('vvr')
def preperat(request):
    return render(request, 'DPU/Otobrazhenie.html')
def naz(request):
    return render(request, 'DPU/Otobrazhenie1.html')

def preperat_i(request):
    nn=statistica1.objects.filter(preparat1=1).count()
    context = {'Podstanovka': statistica1.objects.filter(preparat1=21), 'bb':nn }
    return render(request, 'DPU/Otobrazhenie_statistici.html', context )

def preperat_i1(request):
    context = {'Podstanovka': statistica1.objects.filter(preparat1=2) }
    return render(request, 'DPU/Otobrazhenie_statistici.html', context )
def preperat_i2(request):
    context = {'Podstanovka': statistica1.objects.filter(preparat1=3) }
    return render(request, 'DPU/Otobrazhenie_statistici.html', context )
def preperat_i4(request):
    context = {'Podstanovka': statistica1.objects.filter(preparat1=4) }
    return render(request, 'DPU/Otobrazhenie_statistici.html', context )
def preperat_i15(request):
    context = {'Podstanovka': statistica1.objects.filter(preparat1=5) }
    return render(request, 'DPU/Otobrazhenie_statistici.html', context )
def naz_i(request):
    context = {'Podstanovka': statistica1.objects.filter(diagnoz1=1) }
    return render(request, 'DPU/Otobrazhenie_statistici1.html', context)

```

```

def naz_i1(request):
    context = {'Podstanovka': statistica1.objects.filter(diagnoz1=2) }
    return render(request, 'DPU/Otobrazhenie_statistici1.html', context)
def naz_i2(request):
    context = {'Podstanovka': statistica1.objects.filter(diagnoz1=3) }
    return render(request, 'DPU/Otobrazhenie_statistici1.html', context)
def naz_i3(request):
    context = {'Podstanovka': statistica1.objects.filter(diagnoz1=4) }
    return render(request, 'DPU/Otobrazhenie_statistici1.html', context)
def naz_i4(request):
    context = {'Podstanovka': statistica1.objects.filter(diagnoz1=7) }
    return render(request, 'DPU/Otobrazhenie_statistici1.html', context)

class RegisterUser( CreateView):

    form_class=RegisterUserForm
    template_name = ' DPU /register.html'

    success_url = reverse_lazy('login')


class LoginUser( LoginView):
    form_class=LoginUserForm
    template_name = ' DPU /login.html'

    def get_success_url(self):
        return reverse_lazy('home')

def logout_user(request):
    logout(request) return
    redirect('home')

```

```

from django.contrib.auth.models import User
from django.db import models
from django.urls import reverse

```

```

class Med(models.Model):
    name = models.CharField(max_length=200,verbose_name='Медпрац')
    def __str__(self):
        return self.name
    class Meta:
        verbose_name = 'Працівник'
        verbose_name_plural = 'Працівник'

```

```

class Month(models.Model):
    month = models.CharField(max_length=200 ,verbose_name='Месяц' )
    def __str__(self):
        return self.name
    class Meta:
        verbose_name = 'Монтс'
        verbose_name_plural = 'Монтс'

```

```

class Chis(models.Model):
    chis = models.CharField(max_length=200,verbose_name='Числа загальні')
    def __str__(self):
        return self.name
    class Meta:
        verbose_name = 'День'
        verbose_name_plural = 'День'

```

```

class Cifri(models.Model):

```

```

Cifri = models.CharField(max_length=200,verbose_name='Граф загальний')
def __str__(self):
    return self.name
class Meta:
    verbose_name = 'Час роботи'
    verbose_name_plural = 'Час роботи'

class MonthA(models.Model):

    month = models.ForeignKey('Month', on_delete=models.PROTECT,
verbose_name='Місяц')
    def __str__(self):
        return str(self.month)

class ChisA(models.Model):
    chis = models.ForeignKey('Chis', on_delete=models.PROTECT,
verbose_name='Числа')
    def __str__(self):
        return str(self.chis)

class CifriA(models.Model):
    grafi = models.ForeignKey('Cifri',
on_delete=models.PROTECT,verbose_name='час' )
    def __str__(self):
        return str(self.grafi)

class grafA(models.Model):
    Month = models.ForeignKey('MonthA', on_delete=models.CASCADE,
default=1,verbose_name='Місяць' )
    ChisA = models.ForeignKey('ChisA', on_delete=models.CASCADE,

```

```

verbose_name='Число' )

    grafa = models.ForeignKey('CifriA', on_delete=models.CASCADE,
verbose_name='Тайм' )


DNAME = models.ForeignKey('Med', on_delete=models.PROTECT, default=1, )


ures = models.ForeignKey(User, on_delete=models.PROTECT, null=True,
verbose_name='клиент')


def __str__(self):
    return str(self.Month)


def get_absolute_url(self):
    return reverse('cat', kwargs={'id': self.pk})


class pocta(models.Model):
    name = models.CharField(max_length=200,verbose_name='Медпрац')


class Diagnoz1(models.Model):
    diagnoz =
models.CharField(max_length=200,verbose_name='Діагноз_адмінський')


def __str__(self):
    return str(self.diagnoz)


class Preparat1(models.Model):
    preparat = models.CharField(max_length=200,
verbose_name='Препарат_админский')


def __str__(self):

```

```

    return str(self.preparat)

def get_absolute_url(self):
    return reverse('pre', kwargs={'si': self.pk})

class statistica1(models.Model):
    Doctor123 = models.ForeignKey(Med, on_delete=models.CASCADE, default=1,
verbose_name='Доктор' )
    us_zapis= models.ForeignKey(User, on_delete=models.PROTECT, null=True,
verbose_name='клиент', blank=True)
    diagnoz1 = models.ForeignKey('Diagnoz1', on_delete=models.CASCADE,
verbose_name='Диагноз' )
    preparat1 = models.ForeignKey('Preparat1', on_delete=models.CASCADE,
verbose_name='Препарат' )

    def __str__(self):
        return str(self.Doctor123)

from django.contrib.auth.forms import UserCreationForm, AuthenticationForm
from django.contrib.auth.models import User

from models import*

class grafForms(forms.ModelForm):
    class Meta:
        model= grafA
        fields=['Month','ChisA','grafa']
        Month = forms.CharField(label="Місяць")
        ChisA = forms.CharField(label="Число")

```

```
grafa = forms.CharField(label="Гач")
```

```
class MonthForms(forms.ModelForm):
```

```
    class Meta:
```

```
        model= MonthA
```

```
        fields=['month',]
```

```
class DayForms(forms.ModelForm):
```

```
    class Meta:
```

```
        model= ChisA
```

```
        fields=['chis',]
```

```
class TimeForms(forms.ModelForm):
```

```
    class Meta:
```

```
        model= CifriA
```

```
        fields=['grafi',]
```

```
class Pismo(forms.ModelForm):
```

```
    class Meta:
```

```
        model = pocta
```

```
        fields = ['name', ]
```

```
class statistica1Forms(forms.ModelForm):
```

```
    class Meta:
```

```
        model= statistica1
```

```
        fields=['Doctor123','diagnoz1','preparat1','us_zapis']
```

```
class RegisterUserForm(UserCreationForm):
```

```
    username = forms.CharField(label='Логин',
```

```

widget=forms.TextInput(attrs={'class': 'form-input'}))
    email = forms.EmailField(label='Email', widget=forms.EmailInput(attrs={'class':
'form-input'}))
    password1 = forms.CharField(label='Пароль',
widget=forms.PasswordInput(attrs={'class': 'form-input'}))
    password2 = forms.CharField(label='Повтор пароля',
widget=forms.PasswordInput(attrs={'class': 'form-input'}))
    class Meta:
        model=User
        fields=('username','email','password1','password2')
class LoginUserForm(AuthenticationForm):

    username = forms.CharField(label='Логин',
widget=forms.TextInput(attrs={'class': 'form-input'}))
    password = forms.CharField(label='Пароль',
widget=forms.PasswordInput(attrs={'class': 'form-input'}))

```