

Міністерство освіти і науки України  
Державний університет «Одеська політехніка»  
Інститут комп'ютерних систем  
Кафедра інформаційних систем

## **КУРСОВА РОБОТА**

з дисципліни «Технології створення програмних продуктів»

за темою

«Система обміну корпоративними даними в медичному закладі Co-med»

Пояснювальна записка до етапів визначення вимог до програмного продукту та  
планування процесів розробки програмного продукту

Виконали:

студенти 3-го курсу

групи АІ-193

Савкунов В. С.,

Дмитрієв Ю. Ю.

Перевірив:

Бабіч М. І.

Одеса-2021

## Анотація

В курсовій роботі розглядається процес створення програмного продукту «Система обміну корпоративними даними в медичному закладі» на основних етапах: визначення вимог до програмного продукту, планування процесів розробки, проектування, конструювання, верифікація, розгортання та валідація програмного продукту.

Робота виконувалась в команді з декількох учасників: Савкунов Вадим та Дмитрієв Юрій.

Тому вміст пояснювальної записки розділів «1 Вимоги до програмного продукту» та «2 Планування процесу розробки програмного продукту» співпадає зі вмістом пояснювальної записки інших учасників проєктної команди. у розділах «Проектування», «Конструювання» та «Верифікація» детальніше описано лише одну частину з урахуванням планів проведених робіт з розділу з описом особливостей конструювання:

- структур даних реляційної в системі керування базами даних PostgreSQL;
- програмних модулів в інструментальному середовищі PHPStorm з використанням фреймворку Laravel та мови програмування PHP для backend частини;
- програмних модулів в інструментальному середовищі WebStorm з використанням фреймворку React та мови програмування JavaScript для frontend частини;

Поточну версію пояснювальної записки до результатів роботи розміщено на *GitHub*-репозиторії за адресою: [https://github.com/yurdmित्रiev/co-med\\_application](https://github.com/yurdmित्रiev/co-med_application)

## **Перелік скорочень**

ОС – операційна система

ІС – інформаційна система

БД – база даних

СКБД – система керування базами даних

ПЗ – програмне забезпечення

ПП– програмний продукт

UML – уніфікована мова моделювання

JS — мова програмування JavaScript

ORM — Object Relation Model

MVC — паттерн проектування “Model-View-Controller”

SPA — односторінковий веб-застосунок (Single Page Application)

## Зміст

	стор.
1 Вимоги до програмного продукту	6
1.1 Визначення потреб споживача	6
1.1.1 Ієрархія потреб споживача	6
1.1.2 Деталізація матеріальної потреби	7
1.2 Бізнес-вимоги до програмного продукту	7
1.2.1 Опис проблеми споживача	7
1.2.1.1 Концептуальний опис проблеми споживача	7
1.2.1.2 Опис цільової групи споживача	7
1.2.1.3 Метричний опис проблеми споживача	8
1.2.2 Мета створення програмного продукту	11
1.2.2.1 Проблемний аналіз існуючих програмних продуктів	11
1.2.2.2 Мета створення програмного продукту	12
1.2.3 Назва програмного продукту	12
1.2.3.1 Гасло програмного продукту	12
1.2.3.2 Логотип програмного продукту	12
1.3 Вимоги користувача до програмного продукту	12
1.3.1 Пригодницька історія користувача програмного продукту	12
1.3.2 Історія користувача програмного продукту	13
1.3.3 Діаграма прецедентів програмного продукту	14
1.3.4 Сценаріїв використання прецедентів програмного продукту	14

1.4 Функціональні вимоги до програмного продукту	20
1.4.1. Багаторівнева класифікація функціональних вимог	20
1.4.2 Функціональний аналіз існуючих програмних продуктів	20
1.5 Нефункціональні вимоги до програмного продукту	21
1.5.1 Опис зовнішніх інтерфейсів	21
1.5.1.1 Опис інтерфейсів користувача	22
1.5.1.1.1 Опис INPUT-інтерфейсів користувача	22
1.5.1.1.2 Опис OUTPUT-інтерфейсів користувача	22
1.5.1.2 Опис інтерфейсу із зовнішніми пристроями	29
1.5.1.3 Опис програмних інтерфейсів	29
1.5.1.4 Опис інтерфейсів передачі інформації	29
1.5.1.5 Опис атрибутів продуктивності	29
2 Планування процесу розробки програмного продукту	31
2.1 Планування ітерацій розробки програмного продукту	31
2.2 Концептуальний опис архітектури програмного продукту	32
2.3 План розробки програмного продукту	32
2.3.1 Оцінка трудомісткості розробки програмного продукту	32
2.3.2 Визначення дерева робіт з розробки програмного продукту	35
2.3.3 Графік робіт з розробки програмного продукту	37
2.3.3.1 Таблиця з графіком робіт	38
2.3.3.2 Діаграма Ганта	39

3	Проектування програмного продукту	42
3.1	Концептуальне та логічне проектування структур даних програмного продукту	42
3.1.1	Концептуальне проектування на основі UML-діаграми концептуальних класів	42
3.1.2	Логічне проектування структур даних	43
3.2	Проектування програмних класів	44
3.3	Проектування алгоритмів роботи методів програмних класів	44
3.4	Проектування тестових сценаріїв верифікації роботи програмних модулів	45
3.4.1	Проектування тестових сценаріїв верифікації функціональних вимог	45
3.4.2	Проектування тестових сценаріїв верифікації нефункціональних вимог	45
4	Конструювання програмного продукту	46
4.1	Особливості конструювання структур даних	46
4.2	Особливості конструювання програмних модулів	46
4.2.1	Конструювання алгоритмів методів програмних класів або процедур/функцій	47
4.3	Модульне тестування програмних модулів	52
5	Верифікація програмного продукту	54
5.1	Тестування апаратно-програмних інтерфейсів програмного продукту	54
5.2	Тестування інтерфейсу користувача програмного продукту	54

6 Розгортання та валідація програмного продукту	58
6.1 Інструкція з встановлення системного програмного забезпечення	58
6.2 Інструкція з використання програмного продукту	59
6.3 Результати валідації програмного продукту	59
Висновки до курсової роботи	60

# 1 Вимоги до програмного продукту

## 1.1 Визначення потреб споживача

### 1.1.1. Ієрархія потреб споживача

Відомо, що в теорії маркетингу потреби людини можуть бути представлені у вигляді ієрархії потреб ідей американського психолога Абрахама Маслоу включають рівні:

- фізіологія (вода, їжа, житло, сон);
- безпека (особиста, здоров'я, стабільність),
- приналежність (спілкування, дружба, любов),
- визнання (повага оточуючих, самооцінка),
- самовираження (вдосконалення, персональний розвиток).

На рисунку 1.1 представлено одну ієрархію потреби споживача, яку хотілося б задовольнити, використовуючи майбутній програмний продукт.



Рисунок 1.1.1 – Приклад ієрархії потреби споживача

### 1.1.2 Деталізація матеріальної потреби



Наведено Mindmap-карту деталізації ієрархії потреб споживачів.

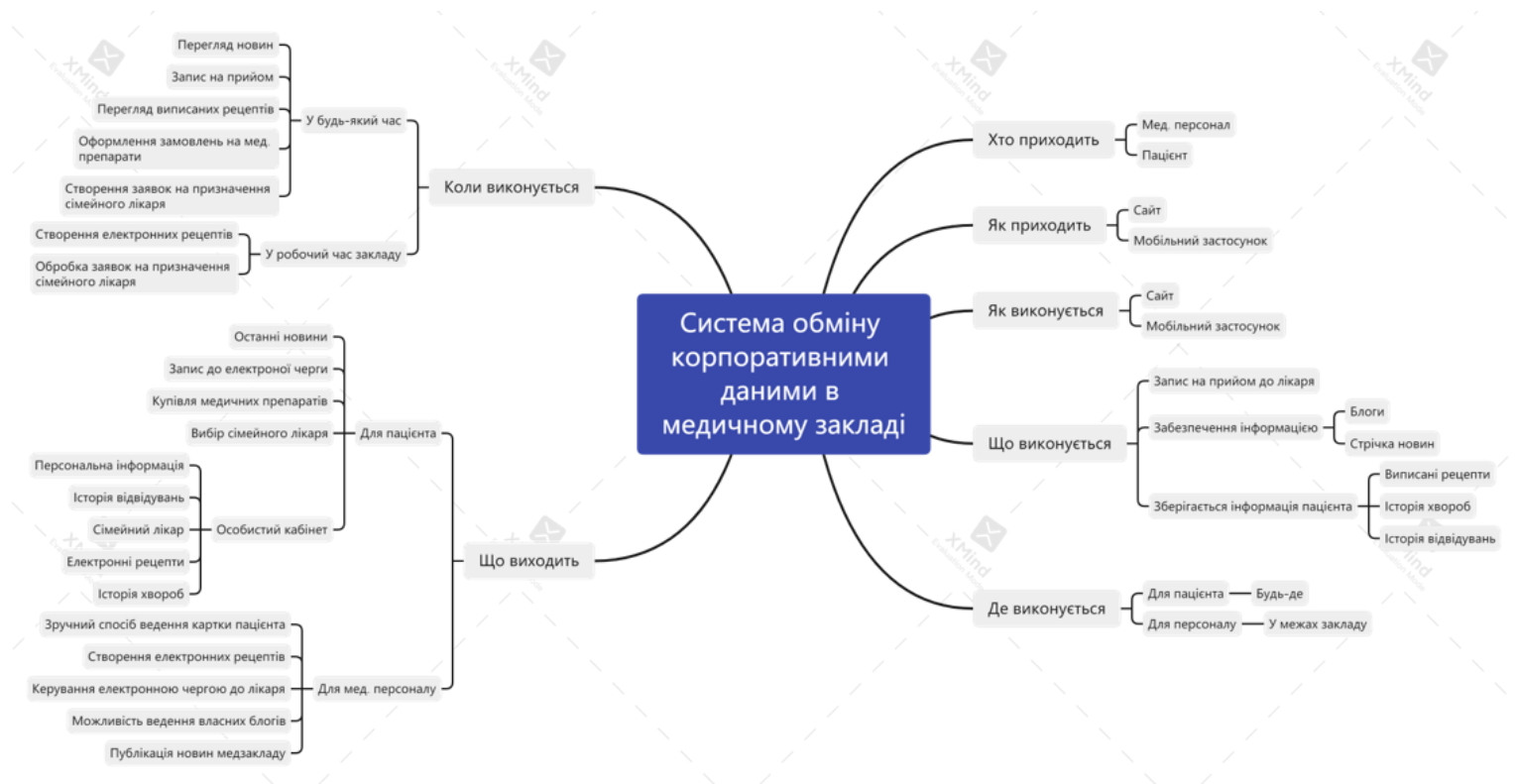


Рис. 1.1.2 – Mindmap-карта деталізації ієрархії потреби споживача

## 1.2 Бізнес-вимоги до програмного продукту

### 1.2.1 Опис проблеми споживача

#### 1.2.1.1 Концептуальний опис проблеми споживача

Неможливо отримувати інформацію та виконувати деякі операції через те, що у клієнтів немає можливості бути присутнім у медичному закладі.

#### 1.2.1.2 Опис цільової групи споживачів

Цільова група споживачів складається з працівників медичного закладу, пацієнтів та зацікавлених у публічній інформації закладу осіб.

### 1.2.1.3 Метричний опис проблеми споживача

Було проведено опитування серед 19 осіб. Опитувані мали відповісти на 6 питань, які допоможуть визначити проблему споживача.

Як часто Ви відвідуєте медичні заклади?

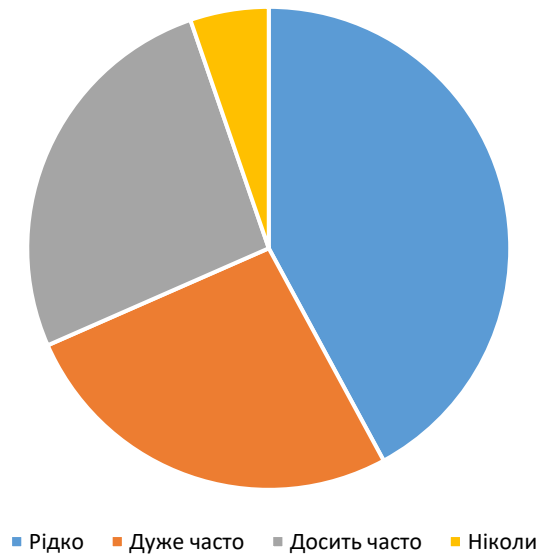


Рис. 1.2.1 - Статистика відповідей на питання «Як часто Ви відвідуєте медичні заклади?»

Як довго Ви чекали своєї черги до лікаря?

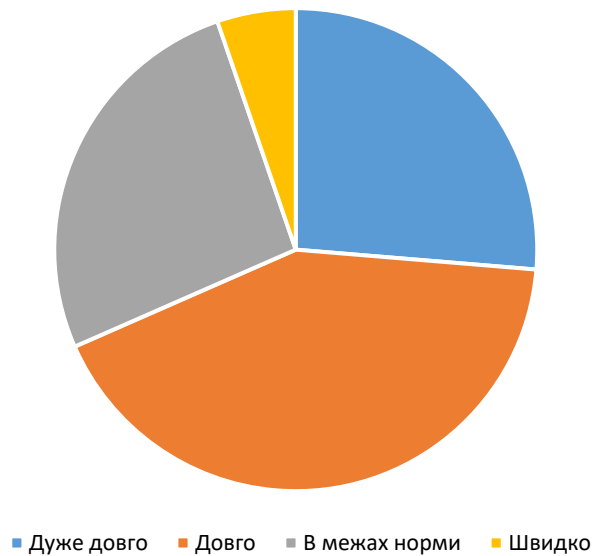


Рис. 1.2.2 - Статистика відповідей на питання «Як довго Ви чекали своєї черги до лікаря?»

Чи готові Ви встати раніше, прийти і зайняти чергу до лікаря?



Рис. 1.2.3 - Статистика відповідей на питання «Чи готові Ви встати раніше, прийти і зайняти чергу до лікаря?»

Чи була в Вас така ситуація: Ви прийшли за аналізами, але виявилося, що вони ще не готові?

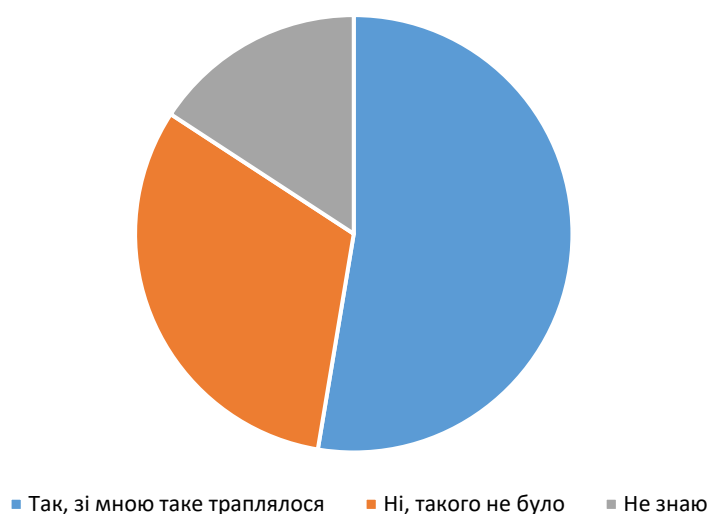


Рис. 1.2.4 - Статистика відповідей на питання «Чи була в Вас така ситуація: Ви прийшли за аналізами, але виявилось, що вони ще не готові?»

Чи є у Вас можливість відвідувати медичні заклади?

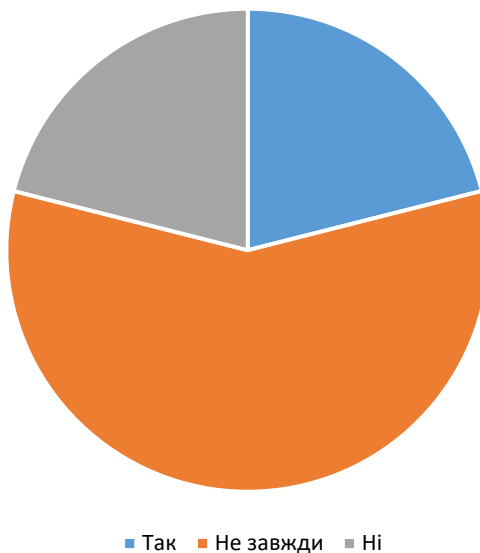


Рис. 1.2.5 - Статистика відповідей на питання «Чи є у Вас можливість відвідувати медичні заклади?»

Чи втрачаєте Ви рецепти виписані лікарем?

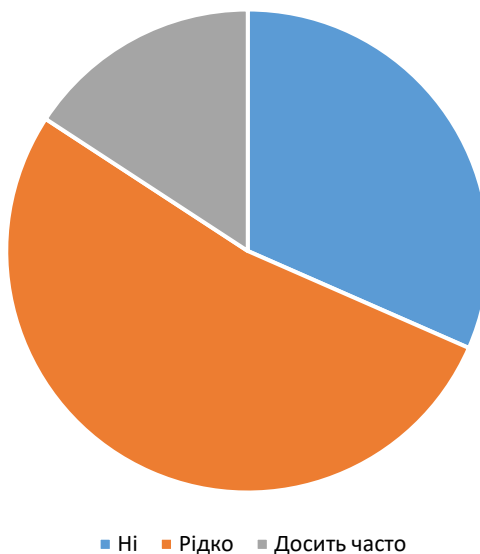


Рис. 1.2.6 - Статистика відповідей на питання «Чи втрачає Ви рецепти  
виписані лікарем?»»

За результатами опитування можна зробити висновок, що більшість опитуваних не має можливості часто відвідувати медичний заклад, тому інформація медичної установи має низьку доступність.

Низький рівень доступності до інформації медичного закладу.

Рівень доступності AL (AL – Access Level) можна визначити як

$$AL = NA / N,$$

де NA - кількість клієнтів медичного закладу, які мають можливість часто бути присутніми у клініці;

N – загальна кількість клієнтів закладу

## 1.2.2 Мета створення програмного продукту

### 1.2.2.1 Проблемний аналіз існуючих програмних продуктів

Таблиця 1.2.2 - Проблемний аналіз існуючих програмних продуктів

<b>Функціонал\Інформаційна система</b>	<b>OnClinic</b>	<b>Odrex</b>	<b>Helsi</b>
Безкоштовність продукту	-	-	+
Ведення блогів	+	+	-
Електронна черга до лікаря	+	+	+
Вибір сімейного лікаря	-	-	-
Перегляд останніх новин	+	+	+
Персональний кабінет пацієнта	-	+	+
Купівля медичних препаратів	-	+	+

Ступінь готовності	1	2	3
--------------------	---	---	---

#### 1.2.2.2 Мета створення програмного продукту

Забезпечення постійного доступу до загальнодоступної та індивідуальної для пацієнта інформації медичного закладу. Надання клієнтам мед. закладу можливості дистанційного виконання деяких операцій.

#### 1.2.3 Назва програмного продукту

Co-Med

##### 1.2.3.1 Гасло програмного продукту

Co-Med - зручне лікування разом.

##### 1.2.3.2 Логотип програмного продукту



рис. 1.2.7 Логотип програмного продукту

#### 1.3 Вимоги користувача до програмного продукту

1.3.1 Пригодницька історія користувача програмного продукту (за бажанням)

Лікарі лікують пацієнтів. Якщо у вас головний чи зубний біль, біль у попереку або болить шлунок чи вухо, болить око або палець, якщо ви відчуваєте

біль у будь-якій частині тіла, якщо боляче рухатися — вам потрібна допомога. Останній раз я навідувався до лікаря офтальмолога. Це лікар, який перевіряє зір у людей. Я записався до онлайн черги на потрібний мені час. Але перед візитом я все одно хвилювався через низький рейтинг лікаря. Останнім часом в мене погіршився зір, можливо тому, що сиджу за комп'ютером занадто довго. Я хвилювався, щоб мені не виписали окуляри. Адже, як мені здається, це буде не зручно. Я маю багато знайомих які мають окуляри. Але я не хочу до них приєднуватися. На щастя лікар сказав, що з моїм зором все добре і виписав електронний рецепт.

### 1.3.2 Історія користувача програмного продукту

Гість:

1. Бути в курсі останніх новин медичного закладу
2. Читає цікаву інформацію з блогів лікарів

Пацієнт:

1. Записується до електронної черги
2. Має доступ до особистої картки пацієнта, а саме:
  - 2.1 Історія хвороб
  - 2.2 Історія відвідувань лікарів
  - 2.3 Електронні рецепти
3. Може обрати сімейного лікаря

Лікар:

1. Заповнює картку пацієнта
2. Виписує електронні рецепти
3. Має можливість вести власний блог

Медичний персонал:

1. Веде стрічку новин

2. Керує електронними чергами до лікарів

### 1.3.3 Діаграма прецедентів програмного продукту

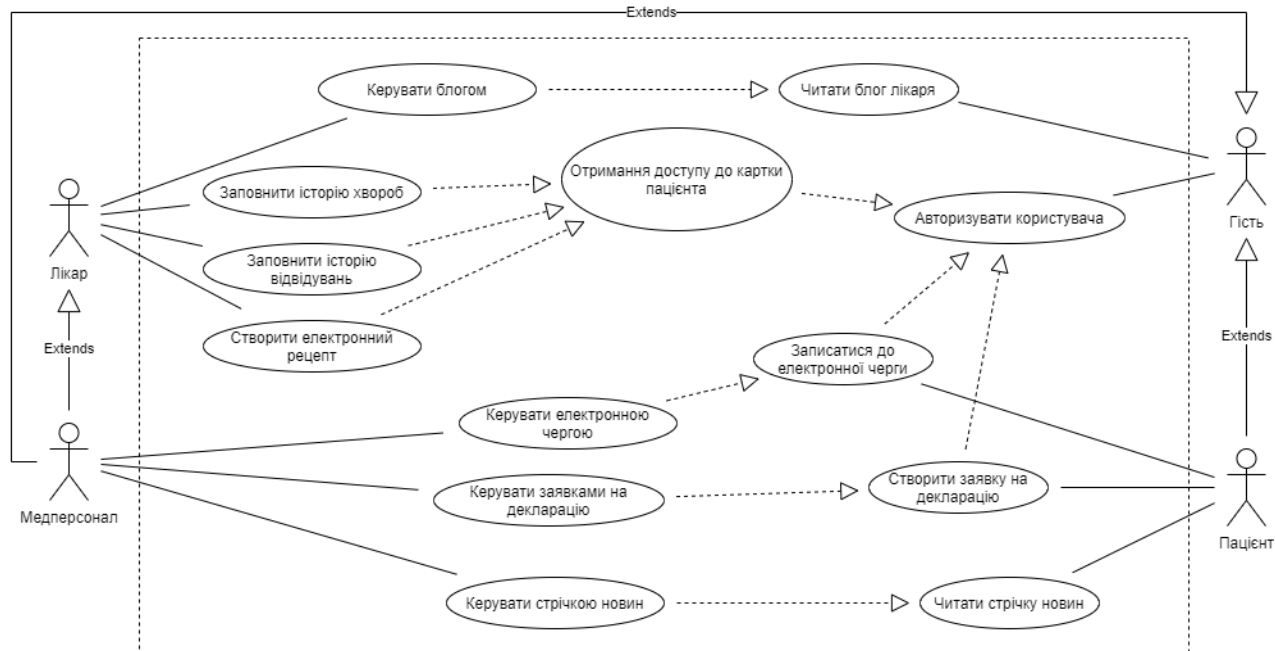


рис. 1.3.1 Діаграма прецедентів

### 1.3.4 Опис сценаріїв використання прецедентів програмного продукту

Назва прецеденту: авторизувати користувача

Передумови початку виконання прецеденту: запуск ПП

Ектори: гість

Ініціатор: гість

Гарантії успіху: гостя успішно авторизовано в системі

Основний успішний сценарій:

1. ПП пропонує гостю заповнити форму авторизації
2. Гість надає свої дані для входу
3. ПП надає доступ до інших прецедентів, які вимагають авторизації

Альтернативний сценарій:

- 3.1. ПП визначив, що дані для входу невірні



### 3.1.1. ПП видає повідомлення про помилку

Назва прецеденту: керувати блогом

Передумови початку виконання прецеденту: лікар пройшов прецедент авторизації

Ектори: лікар

Ініціатор: лікар

Гарантії успіху: лікар отримав список записів блогу та може ними керувати

Основний успішний сценарій:

1. ПП пропонує перейти у розділ керування власним блогом
2. Лікар переходить у відповідний розділ ПП
3. ПП виводить список записів блогу та панель керування, яка дає можливість видаляти, змінювати та додавати записи

Назва прецеденту: читати блог лікаря

Передумови початку виконання прецеденту: запуск ПП

Ектори: гість

Ініціатор: гість

Гарантії успіху: гість отримав текст запису

Основний успішний сценарій:

1. ПП виводить список записів блогу
2. Гість обирає одну статтю зі списку
3. ПП виводить повний текст статті

Назва прецеденту: керувати стрічкою новин

Передумови початку виконання прецеденту: медперсонал пройшов прецедент авторизації

Ектори: медперсонал

Ініціатор: медперсонал

Гарантії успіху: медперсонал отримав список створених новин та може ними керувати

Успішний сценарій:

1. ПП пропонує перейти у розділ керування новинами
2. Медперсонал переходить у відповідний розділ ПП
3. ПП виводить список записів блогу та панель керування, яка дає можливість видаляти, змінювати та додавати записи

Назва прецеденту: читати стрічку новин

Передумови початку виконання прецеденту: запуск ПП

Ектори: гість

Ініціатор: гість

Гарантії успіху: гість отримав текст новини

Успішний сценарій:

1. ПП виводить список новин
2. Гість обирає новину зі списку
3. ПП виводить повний текст новини

Назва прецеденту: отримати доступ до картки пацієнта

Передумови початку виконання прецеденту: пацієнт пройшов прецедент авторизації

Ектори: пацієнт

Ініціатор: пацієнт

Гарантії успіху: пацієнт отримує всю інформацію, яка міститься у його амбулаторній картці

Основний успішний сценарій:

1. ПП пропонує обрати один з розділів картки
2. Пацієнт переходить за обраним посиланням
3. ПП виводить усю інформацію з обраного розділу

Назва прецеденту: створити електронний рецепт

Передумови початку виконання прецеденту: лікар пройшов прецедент авторизації

Ектори: лікар

Ініціатор: лікар

Гарантії успіху: лікар може виписати електронний рецепт

Основний успішний сценарій:

1. ПП пропонує відкрити карту пацієнта
2. Лікар переходить за посиланням «Новий рецепт»
3. ПП виводить форму створення рецепта
4. Лікар заповнює форму та зберігає введені дані

Назва прецеденту: заповнити історію відвідувань

Передумови початку виконання прецеденту: лікар пройшов прецедент авторизації

Ектори: лікар

Ініціатор: лікар

Гарантії успіху: лікар може зробити відмітку про візит пацієнта

Основний успішний сценарій:

1. ПП пропонує відкрити карту пацієнта
2. Лікар переходить за посиланням «Зафіксувати візит»
3. ПП виводить форму нового візиту
4. Лікар заповнює форму та зберігає введені дані

Назва прецеденту: заповнити історію хвороб

Передумови початку виконання прецеденту: лікар пройшов прецедент авторизації

Ектори: лікар

Ініціатор: лікар

Гарантії успіху: лікар може зробити відмітку про хвороби пацієнта

Основний успішний сценарій:

1. ПП пропонує відкрити карту пацієнта
2. Лікар переходить за посиланням «Додати відомості про хворобу»
3. ПП виводить форму нового візиту
4. Лікар заповнює форму та зберігає введені дані

Назва прецеденту: записатися до електронної черги

Передумови початку виконання прецеденту: пацієнт пройшов прецедент авторизації

Ектори: пацієнт

Ініціатор: пацієнт

Гарантії успіху: пацієнт отримав повідомлення про те, що він записаний до лікаря із можливістю роздрукувати талон.

Основний успішний сценарій:

1. ПП пропонує список лікарів, до яких можна записатися
2. Пацієнт обирає лікаря
3. ПП пропонує вільний час, який можна зарезервувати
4. Пацієнт обирає зручний час візиту із запропонованих

Альтернативний сценарій:

- 3.1. ПП не може запропонувати час для запису, бо всі місця зарезервовані

Назва прецеденту: керувати електронною чергою

Передумови початку виконання прецеденту: медперсонал пройшов прецедент авторизації

Ектори: медперсонал

Ініціатор: медперсонал

Гарантії успіху: медперсонал може керувати чергою до лікаря

Основний успішний сценарій:

1. ПП пропонує список людей та лікарів, до кого вони записались, та панель керування для виконання дій над записами

2. Медперсонал обирає один із записів
3. ПП пропонує виконати одну з дій: видалити, редагувати запис

Назва прецеденту: створити заявку на декларацію

Передумови початку виконання прецеденту: пацієнт пройшов прецедент авторизації

Ектори: пацієнт

Гарантії успіху: пацієнт отримує повідомлення, що заявку успішно надіслано

Успішний сценарій:

1. ПП пропонує список лікарів, з якими можна укласти декларацію
2. Пацієнт обирає лікаря
3. Пацієнт надсилає заявку
4. ПП зберігає заявку та виводить повідомлення про успішне виконання операції

Назва прецеденту: керувати заявками на декларацію

Передумови початку виконання прецеденту: медперсонал пройшов прецедент авторизації

Ектори: медперсонал

Ініціатор: медперсонал

Гарантії успіху: медперсонал надіслав пацієнту повідомлення про результат обробки заявки

Основний успішний сценарій:

1. ПП виводить список необроблених заявок
2. Медперсонал обирає одну із заявок
3. Медперсонал оброблює заявку
4. ПП надсилає повідомлення пацієнту про результат обробки заявки

#### 1.4 Функціональні вимоги до програмного продукту

#### 1.4.1. Багаторівнева класифікація функціональних вимог

Таблиця 1.4.1 - Багаторівнева класифікація функціональних вимог

Ідентифікатор функції	Назва функції
FR1	Блоги
FR1.1	Створення записів блогу
FR1.2	Перегляд блогів
FR2	Новини
FR2.1	Створення новин
FR2.2	Перегляд новин
FR3	Персональний кабінет
FR3.1	Картка пацієнта
FR3.1.1	Історія хвороб
FR3.1.2	Історія відвідувань
FR3.1.3	Електронні рецепти
FR3.2	Запис до лікаря
FR3.3	Вибір сімейного лікаря
FR4	Авторизація

#### 1.4.2 Функціональний аналіз існуючих програмних продуктів

Таблиця 1.4.2 - Функціональний аналіз існуючих програмних продуктів

Ідентифікатор функції	ПП1 (OnClinic)	ПП2 (Odrex)	ПП3 (Helsi)
FR1	-	+	-
FR1.1	-	+	-

FR1.2	-	+	-
FR2	+	+	+
FR2.1	+	+	+
FR2.2	+	+	+
FR3	+	+	+
FR3.1	+	+	+
FR3.1.1	-	+	+
FR3.1.2	+	+	+
FR3.1.3	-	-	+
FR3.2	+	+	+
FR3.3	-	+	+
FR4	+	+	+

## 1.5 Нефункціональні вимоги до програмного продукту

### 1.5.1 Опис зовнішніх інтерфейсів

Таблиця 1.5.1 - Опис зовнішніх інтерфейсів

Ідентифікатор функції	Зовнішній пристрій
FR1.1	Персональний комп'ютер, смартфон, планшет
FR1.2	
FR2.1	
FR2.2	
FR3.1	
FR3.1.1	
FR3.1.2	

FR3.1.3	
FR3.2	
FR3.3	
FR4	

#### 1.5.1.1 Опис інтерфейсів користувача

##### 1.5.1.1.1 Опис INPUT-інтерфейсів користувача

Клавіатура, миша, сенсорний екран

##### 1.5.1.1.2 Опис OUTPUT-інтерфейсів користувача

Функціонал:

- FR1.1 (створення статті блогу),
- FR2.1(створення новини)

Засіб OUTPUT-потoku: графічний інтерфейс



<b>Заголовок</b>	<b>Расширенный заголовок</b>
<input type="text" value="Заголовок"/>	<input type="text" value="Расширенный заголовок"/>
<b>Миниатюра статьи</b>	<b>Ключевые слова</b>
<div><div>200x200</div></div>	<input type="text" value="Тег"/>
Кликните на изображение, чтобы изменить миниатюру	Перечислить ключевые слова через запятую
<b>Содержимое</b>	<input type="checkbox"/> Опубликовать
<div><div>Текст</div></div>	

---

Отмена

Сохранить

Рис. 1.5.1 - Графічний інтерфейс створення статті блогу

Функціонал:

- FR1.2 (перегляд блогу),
- FR2.2 (перегляд новин)

Засіб OUTPUT-потoku: графічний інтерфейс

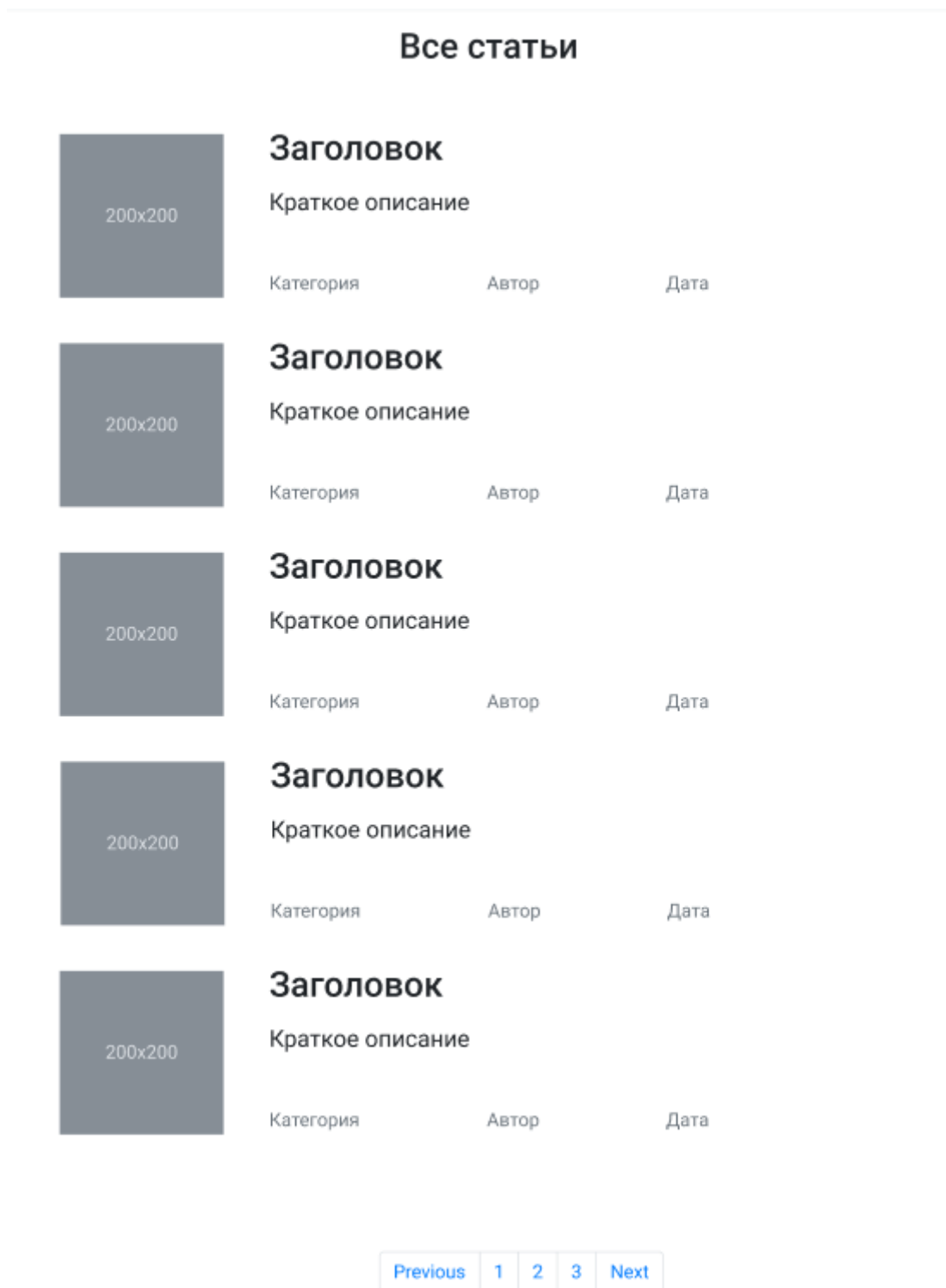


Рис. 1.5.2 - Графічний інтерфейс перегляду блогів

Функціонал: FR3 (персональний кабінет),  
Засіб OUTPUT-поток: графічний інтерфейс

## Личный кабинет

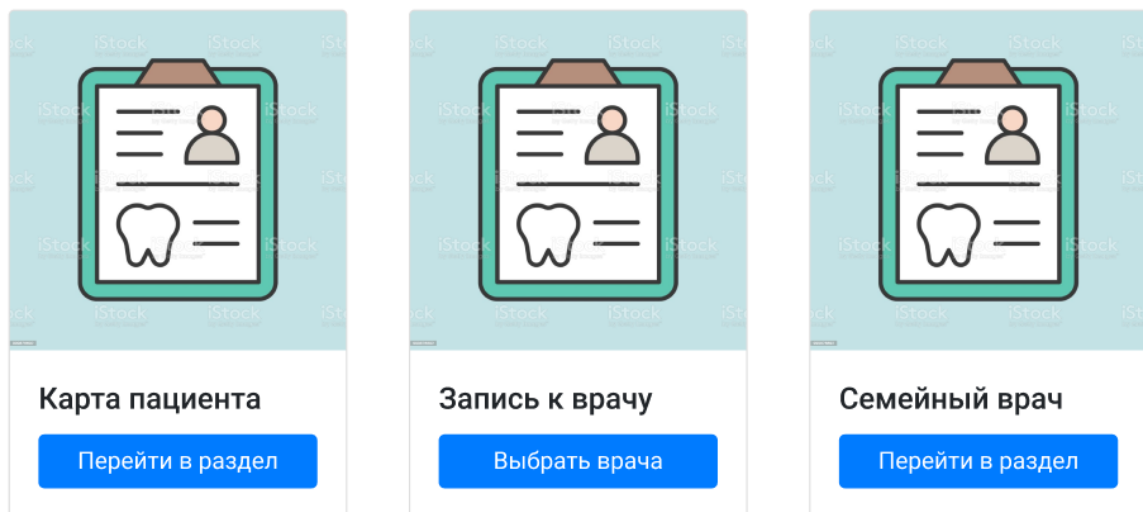


Рис. 1.5.3 - Графічний інтерфейс персонального кабінету

Функціонал: FR3.1.1 (історія хвороб)

Засіб OUTPUT-поток: графічний інтерфейс

### История болезней

Диагноз	Дата установления диагноза	Дата выздоровления	
ОРВИ	15.06.2021	18.06.2021	<a href="#">Подробнее</a>
ОРВИ	15.06.2021	18.06.2021	<a href="#">Подробнее</a>
ОРВИ	15.06.2021	18.06.2021	<a href="#">Подробнее</a>

Рис. 1.5.4 - Графічний інтерфейс історії хвороб

Функціонал: FR3.1.2 (історія відвідувань)

Засіб OUTPUT-поток: графічний інтерфейс

### История посещений

Врач	Время визита	Цель визита	
Пилюлькин А.В.	15.06.2021 15:30	Жалобы на кашель	<a href="#">Подробнее</a>
Пилюлькин А.В.	15.06.2021 15:30	Жалобы на кашель	<a href="#">Подробнее</a>
Пилюлькин А.В.	15.06.2021 15:30	Жалобы на кашель	<a href="#">Подробнее</a>
Пилюлькин А.В.	15.06.2021 15:30	Жалобы на кашель	<a href="#">Подробнее</a>

Рис. 1.5.5 - Графічний інтерфейс історії відвідувань

Функціонал: FR3.1.3 (електронні рецепти)

Засіб OUTPUT-поток: графічний інтерфейс

### Список рецептов

Идентификатор	Дата приписания	Кто выписал	
№12345678	15.06.2021	Пилюлькин А.В.	<a href="#">Подробнее</a>
№12345678	15.06.2021	Пилюлькин А.В.	<a href="#">Подробнее</a>
№12345678	15.06.2021	Пилюлькин А.В.	<a href="#">Подробнее</a>
№12345678	15.06.2021	Пилюлькин А.В.	<a href="#">Подробнее</a>

Рис. 1.5.6 - Графічний інтерфейс електронних рецептів

Функціонал: FR3.2 (запис на прийом)

Засіб OUTPUT-поток: графічний інтерфейс

### Запись на приём

Поиск

200x200	<b>Пилюлькин А.В.</b> Педиатр	Пн-Сб 10:00 — 21:00	<a href="#" style="border: 1px solid #007bff; padding: 5px 10px; color: #007bff;">Выбрать время</a>
200x200	<b>Пилюлькин А.В.</b> Педиатр	Пн-Сб 10:00 — 21:00	<a href="#" style="border: 1px solid #007bff; padding: 5px 10px; color: #007bff;">Выбрать время</a>
200x200	<b>Пилюлькин А.В.</b> Педиатр	Пн-Сб 10:00 — 21:00	<a href="#" style="border: 1px solid #007bff; padding: 5px 10px; color: #007bff;">Выбрать время</a>

Рис. 1.5.7 - Графічний інтерфейс запису на прийом

Функціонал: FR3.3 (вибір сімейного лікаря)

Засіб OUTPUT-поток: графічний інтерфейс

### Выбор семейного врача

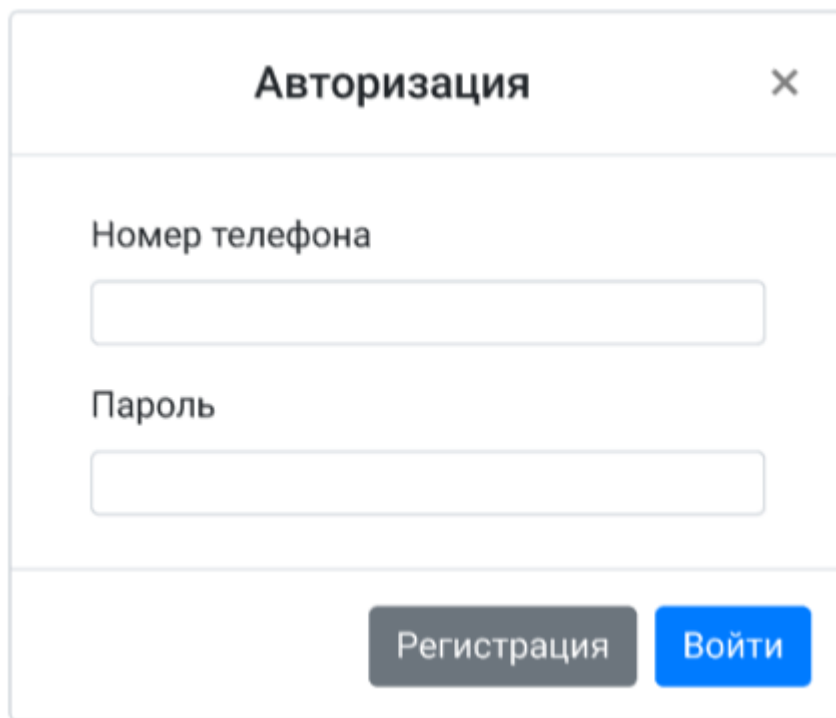
Поиск

200x200	<b>Пилюлькин А.В.</b> Педиатр	Пн-Сб 10:00 — 21:00	<a href="#" style="border: 1px solid #007bff; padding: 5px 10px; color: #007bff;">Подать заявку</a>
200x200	<b>Пилюлькин А.В.</b> Педиатр	Пн-Сб 10:00 — 21:00	<a href="#" style="border: 1px solid #007bff; padding: 5px 10px; color: #007bff;">Подать заявку</a>
200x200	<b>Пилюлькин А.В.</b> Педиатр	Пн-Сб 10:00 — 21:00	<a href="#" style="border: 1px solid #007bff; padding: 5px 10px; color: #007bff;">Подать заявку</a>

Рис. 1.5.8 - Графічний інтерфейс вибору сімейного лікаря

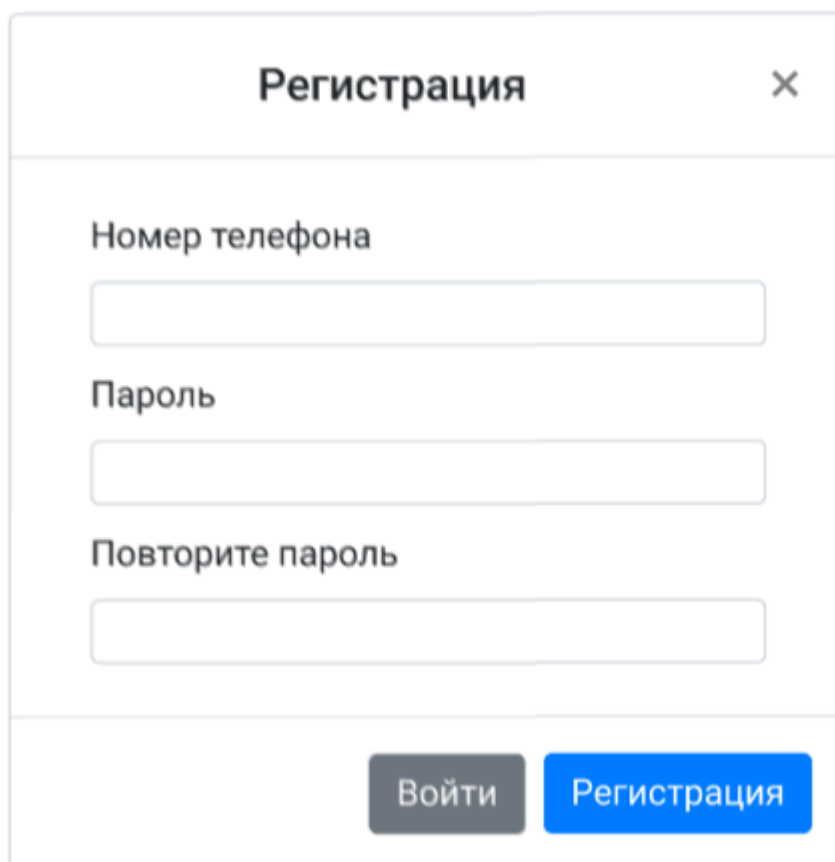
Функціонал: FR4 (авторизація)

Засіб OUTPUT-поток: графічний інтерфейс



The image shows a web form titled "Авторизация" (Authorization) with a close button (X) in the top right corner. The form is divided into three horizontal sections. The first section contains the label "Номер телефона" (Phone number) above a text input field. The second section contains the label "Пароль" (Password) above another text input field. The third section contains two buttons: a grey button labeled "Регистрация" (Registration) and a blue button labeled "Войти" (Login).

Рис. 1.5.9 - Графічний інтерфейс авторизації



The image shows a web form titled "Регистрация" (Registration) with a close button (X) in the top right corner. The form is divided into three horizontal sections. The first section contains the label "Номер телефона" (Phone number) above a text input field. The second section contains the label "Пароль" (Password) above a text input field. The third section contains the label "Повторите пароль" (Repeat password) above a text input field. The bottom section contains two buttons: a grey button labeled "Войти" (Login) and a blue button labeled "Регистрация" (Registration).

Рис. 1.5.10 - Графічний інтерфейс реєстрації

#### 1.5.1.2 Опис інтерфейсу із зовнішніми пристроями

Не використовуються зовнішні пристрої

#### 1.5.1.3 Опис програмних інтерфейсів

Версії операційних систем та програмних бібліотек, які знадобляться при реалізації більшості функцій ПП:

- ОС Windows/Linux для персональних комп'ютерів, Android/iOS для мобільних пристроїв
- Останні версії браузерів Chrome (та інші браузери на основі Chromium), Firefox, Webkit-браузерів (Safari, Eriphany та ін.) з увімкненим функціоналом JS

#### 1.5.1.4 Опис інтерфейсів передачі інформації

Провідні: Ethernet

Безпроводні: Wi-fi

#### 1.5.1.5 Опис атрибутів продуктивності

Таблиця 1.5.1 - Опис атрибутів продуктивності

Ідентифікатор функції	Максимальний час реакції ПП на дії користувачів, секунди
FR1.1	2
FR1.2	1
FR2.1	2
FR2.2	1
FR3.1	1

FR3.1.1	2
FR3.1.2	2
FR3.1.3	2
FR3.2	3
FR3.3	3
FR4	2



## 2 Планування процесу розробки програмного продукту

### 2.1 Планування ітерацій розробки програмного продукту

З метою забезпечення вимог таких рекомендацій IEEE-стандарту, як необхідність, корисність при експлуатації, здійсненність функціональних вимог до ПП, визначено функціональні пріоритети, які будуть використані при плануванні ітерацій розробки ПП. Результати представлено в таблиці 2.1

Таблиця 2.1 – приклад опису функцій з наданням унікальних ієрархічних ідентифікаторів

Ідентифікатор функції (назва)	Назва функції
FR1	Блоги
FR1.1	Створення записів блогу
FR1.2	Перегляд блогів
FR2	Новини
FR2.1	Створення новин
FR2.2	Перегляд новин
FR3	Персональний кабінет
FR3.1	Картка пацієнта
FR3.1.1	Історія хвороб
FR3.1.2	Історія відвідувань
FR3.1.3	Електронні рецепти
FR3.2	Запис до лікаря
FR3.3	Вибір сімейного лікаря
FR4	Авторизація

## 2.2 Концептуальний опис архітектури програмного продукту

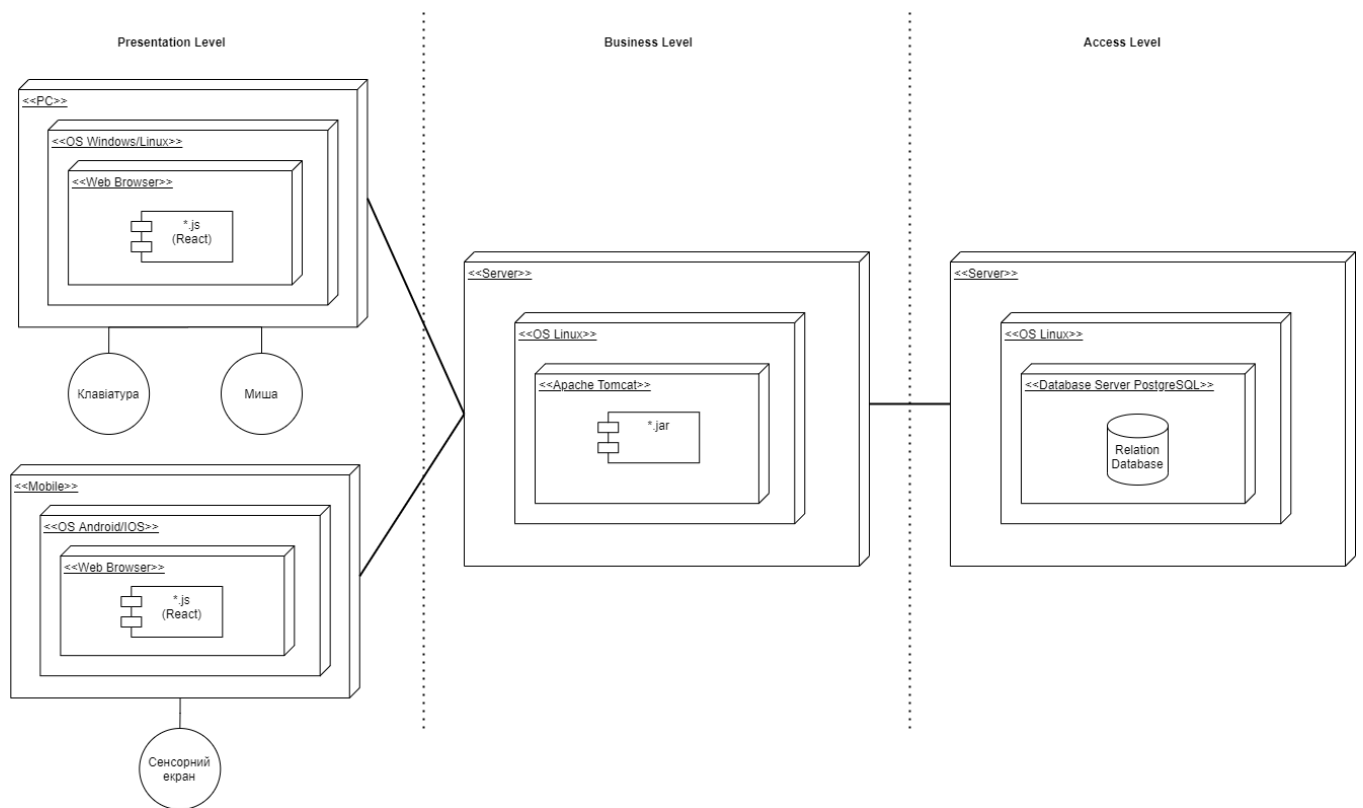


Рис. 2.1 UML-діаграма розгортання

## 2.3 План розробки програмного продукту

### 2.3.1 Оцінка трудомісткості розробки програмного продукту

Таблиця 2.2 – Ваговий коефіцієнт акторів

Актор	Тип Актора	Ваговий коефіцієнт
Пацієнт	Складний	2
Лікар	Середній	3
Медперсонал	Середній	3
Гість	Складний	3

$$A = 11$$

Таблиця 2.3 – Вагові коефіцієнти прецедентів

Прецедент	Тип прецедента	Кількість кроків сценарію	Ваговий коефіцієнт
Авторизувати користувача	Складний	3	15
Керувати блогом	Середній	3	2
Читати блог лікаря	Простий	3	1
Керувати стрічкою новин	Середній	3	2
Читати стрічку новин	Простий	3	1
Отримати доступ до картки пацієнта	Середній	3	10
Створити електронний рецепт	Середній	4	5
Заповнити історію відвідувань	Простий	4	5
Заповнити історію хвороб	Середній	4	5
Записатися до електронної черги	Простий	4	3
Керувати електронною чергою	Середній	3	5
Створити заявку на декларацію	Середній	4	10
Керувати заявками на декларацію	Середній	4	10

UC = 74

UUCP = 85

Таблиця 2.4 – Показники технічної складності проекту

Показник	Значення	Вага
T1	3	2
T2	5	1
T3	5	1
T4	2	-1
T5	1	1
T6	5	0.5
T7	5	0.5
T8	5	2
T9	3	1
T10	3	1
T11	4	1
T12	4	1
T13	2	1

$$TCF = 1.06$$

$$EF = 1.085$$

$$UCP = 85 * 1.06 * 1.085 = 97.7585$$

$$\text{Загальна кількість годин: } 97.7585 * 20 = 1955.17$$

### 2.3.2 Визначення дерева робіт з розробки програмного продукту

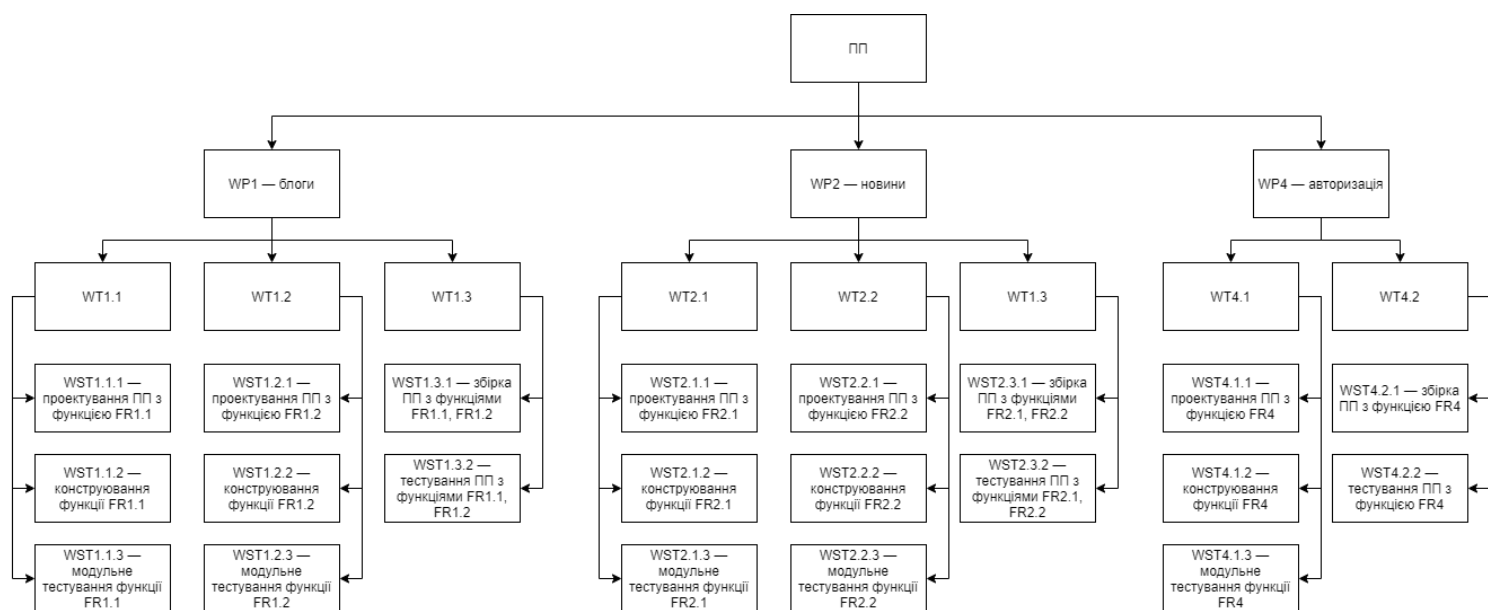


Рис. 2.3.2 - Дерево робіт з розробки програмного продукту

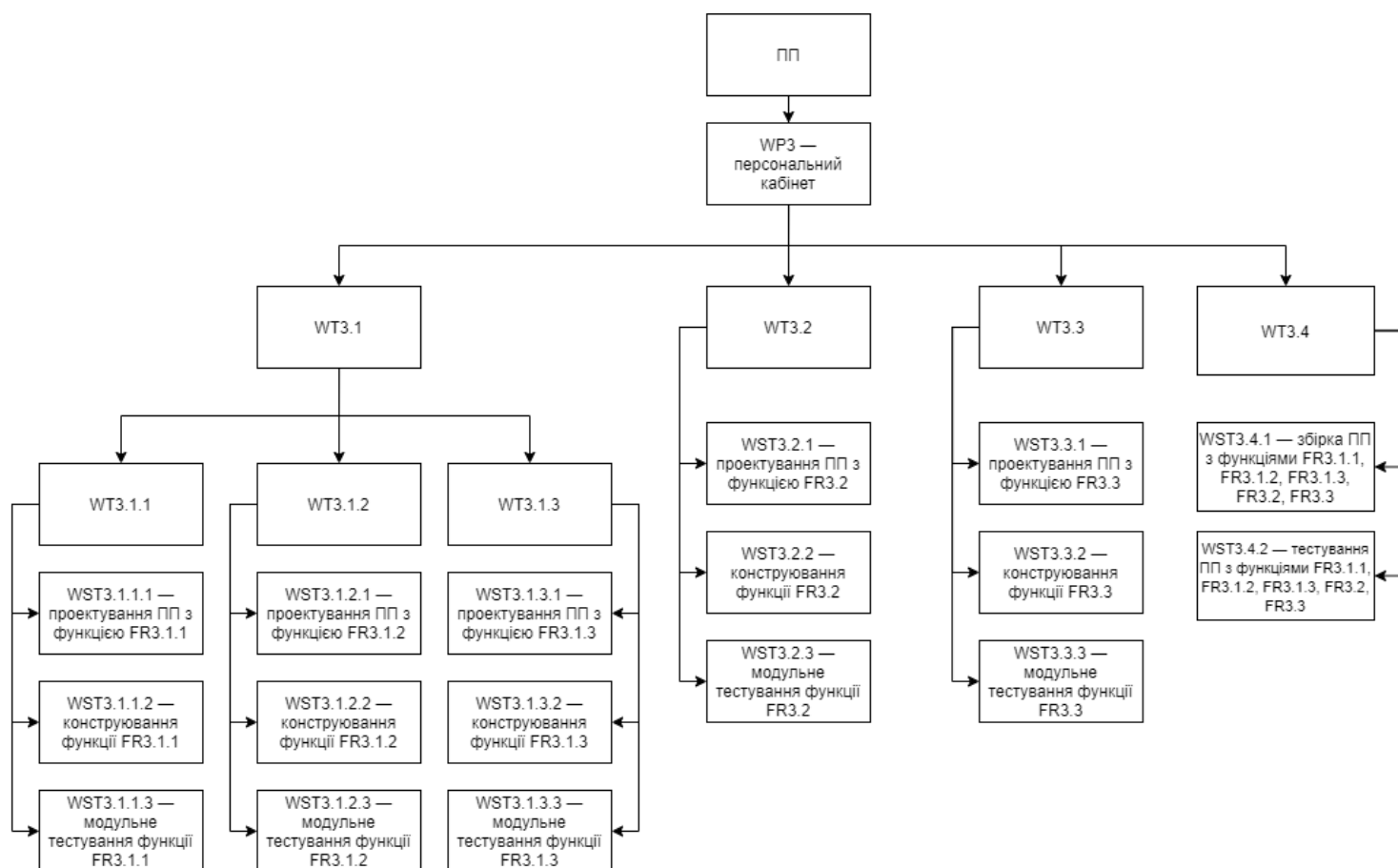


Рис. 2.3.3 - Дерево робіт з розробки програмного продукту

Таблиця 2.3.4 – Опис підзадач із закріпленням виконавців

Підзадача	Виконавець
WST1.1.1	Дмитрієв Ю.Ю.
WST1.1.2	Дмитрієв Ю.Ю.
WST1.1.3	Дмитрієв Ю.Ю.
WST1.2.1	Дмитрієв Ю.Ю.
WST1.2.2	Дмитрієв Ю.Ю.
WST1.2.3	Дмитрієв Ю.Ю.
WST1.3.1	Дмитрієв Ю.Ю.
WST1.3.2	Дмитрієв Ю.Ю.
WST2.1.1	Савкунов. В. С.
WST2.1.2	Савкунов. В. С.
WST2.1.3	Савкунов. В. С.
WST2.2.1	Савкунов. В. С.
WST2.2.2	Савкунов. В. С.
WST2.2.3	Савкунов. В. С.
WST2.3.1	Савкунов. В. С.
WST2.3.2	Савкунов. В. С.
WST3.1.1.1	Савкунов. В. С.
WST3.1.1.2	Савкунов. В. С.
WST3.1.1.3	Савкунов. В. С.
WST3.1.2.1	Савкунов. В. С.
WST3.1.2.2	Савкунов. В. С.
WST3.1.2.3	Савкунов. В. С.
WST3.1.3.1	Савкунов. В. С.
WST3.1.3.2	Савкунов. В. С.
WST3.1.3.3	Савкунов. В. С.
WST3.2.1	Дмитрієв Ю.Ю.

WST3.2.2	Дмитрієв Ю.Ю.
WST3.2.3	Дмитрієв Ю.Ю.
WST3.3.1	Дмитрієв Ю.Ю.
WST3.3.2	Дмитрієв Ю.Ю.
WST3.3.3	Дмитрієв Ю.Ю.
WST3.4.1	Дмитрієв Ю.Ю.
WST3.4.2	Дмитрієв Ю.Ю.
WST4.1.1	Савкунов. В. С.
WST4.1.2	Савкунов. В. С.
WST4.1.3	Савкунов. В. С.
WST4.2.1	Савкунов. В. С.
WST4.2.2	Савкунов. В. С.

### 2.3.3 Графік робіт з розробки програмного продукту

#### 2.3.3.1 Таблиця з графіком робіт

Таблиця 2.3.5 – Графік робіт

WST	Дата початку	Дні	Дата завершення	Виконавець
1.1.1	10.10.2021	2	11.10.2021	Дмитрієв Ю. Ю.
1.1.2	12.10.2021	5	16.10.2021	Дмитрієв Ю. Ю.
1.1.3	17.10.2021	2	18.10.2021	Дмитрієв Ю. Ю.
1.2.1	19.10.2021	2	20.10.2021	Дмитрієв Ю. Ю.
1.2.2	21.10.2021	5	25.10.2021	Дмитрієв Ю. Ю.
1.2.3	26.10.2021	2	27.10.2021	Дмитрієв Ю. Ю.
1.3.1	28.10.2021	1	28.10.2021	Дмитрієв Ю. Ю.
1.3.2	29.10.2021	4	01.11.2021	Дмитрієв Ю. Ю.
2.1.1	10.10.2021	2	11.10.2021	Савкунов. В. С.
2.1.2	12.10.2021	5	16.10.2021	Савкунов. В. С.
2.1.3	17.10.2021	2	18.10.2021	Савкунов. В. С.
2.2.1	19.10.2021	2	20.10.2021	Савкунов. В. С.
2.2.2	21.10.2021	5	25.10.2021	Савкунов. В. С.
2.2.3	26.10.2021	2	27.10.2021	Савкунов. В. С.
2.3.1	28.10.2021	1	28.10.2021	Савкунов. В. С.
2.3.2	29.10.2021	4	01.11.2021	Савкунов. В. С.
3.1.1.1	02.11.2021	3	04.11.2021	Савкунов. В. С.
3.1.1.2	05.11.2021	5	09.11.2021	Савкунов. В. С.
3.1.1.3	10.11.2021	2	11.11.2021	Савкунов. В. С.
3.1.2.1	12.11.2021	3	14.11.2021	Савкунов. В. С.
3.1.2.2	15.11.2021	5	19.11.2021	Савкунов. В. С.
3.1.2.3	20.11.2021	2	21.11.2021	Савкунов. В. С.
3.1.3.1	22.11.2021	3	24.11.2021	Савкунов. В. С.

3.1.3.2	25.11.2021	5	29.11.2021	Савкунов. В. С.
3.1.3.3	30.11.2021	2	01.12.2021	Савкунов. В. С.
3.2.1	02.11.2021	3	04.11.2021	Дмитрієв Ю. Ю.
3.2.2	05.11.2021	10	14.11.2021	Дмитрієв Ю. Ю.
3.2.3	15.11.2021	3	17.11.2021	Дмитрієв Ю. Ю.
3.3.1	18.11.2021	3	20.11.2021	Дмитрієв Ю. Ю.
3.3.2	21.11.2021	10	30.11.2021	Дмитрієв Ю. Ю.
3.3.3	01.12.2021	3	03.12.2021	Дмитрієв Ю. Ю.
3.4.1	04.12.2021	1	04.12.2021	Дмитрієв Ю. Ю.
3.4.2	05.12.2021	5	09.12.2021	Дмитрієв Ю. Ю.
4.1.1	02.12.2021	2	03.12.2021	Савкунов. В. С.
4.1.2	04.12.2021	2	05.12.2021	Савкунов. В. С.
4.1.3	06.12.2021	2	07.12.2021	Савкунов. В. С.
4.2.1	08.12.2021	1	08.12.2021	Савкунов. В. С.
4.2.2	09.12.2021	1	09.12.2021	Савкунов. В. С.



### 2.3.3.2 Діаграма Ганта

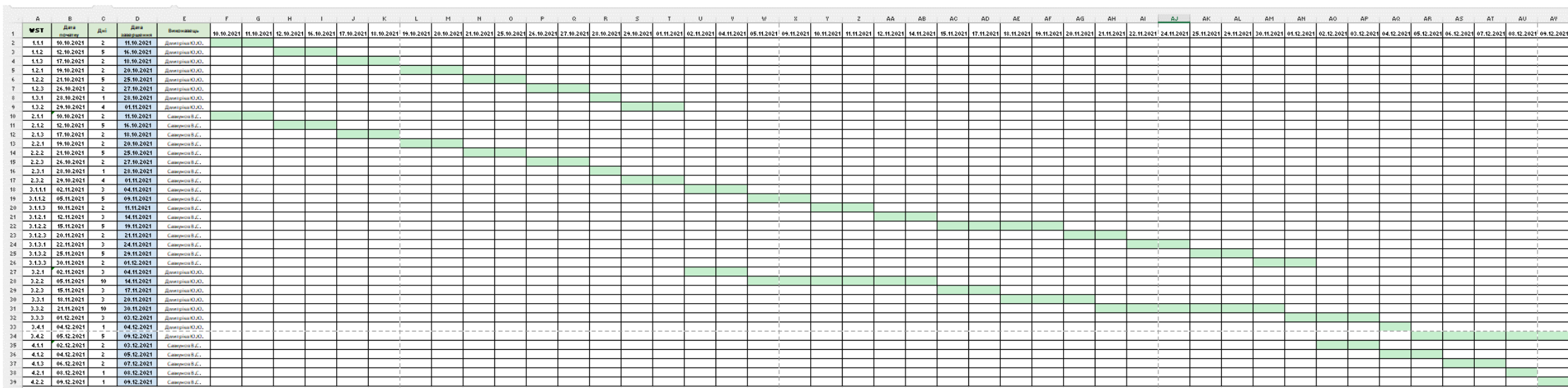


Рис. 2.3.3.2 Діаграма Ганта

Через великий розмір рисунка текст на ньому погано видно. Тому таблицю із діаграмою можна детальніше побачити за посиланням: [https://opuuu-my.sharepoint.com/:x:/g/personal/yurdmित्रiev\\_o365\\_opu\\_ua/ERUCs-B4J2dFplGTfKpiZ-0BoJuFRmsTMVOCwB9LvRv5Zw?e=MXthEv](https://opuuu-my.sharepoint.com/:x:/g/personal/yurdmित्रiev_o365_opu_ua/ERUCs-B4J2dFplGTfKpiZ-0BoJuFRmsTMVOCwB9LvRv5Zw?e=MXthEv)

### 3 Проектування програмного продукту

#### 3.1 Концептуальне та логічне проектування структур даних програмного продукту

##### 3.1.1. Концептуальне проектування на основі UML-діаграми концептуальних класів

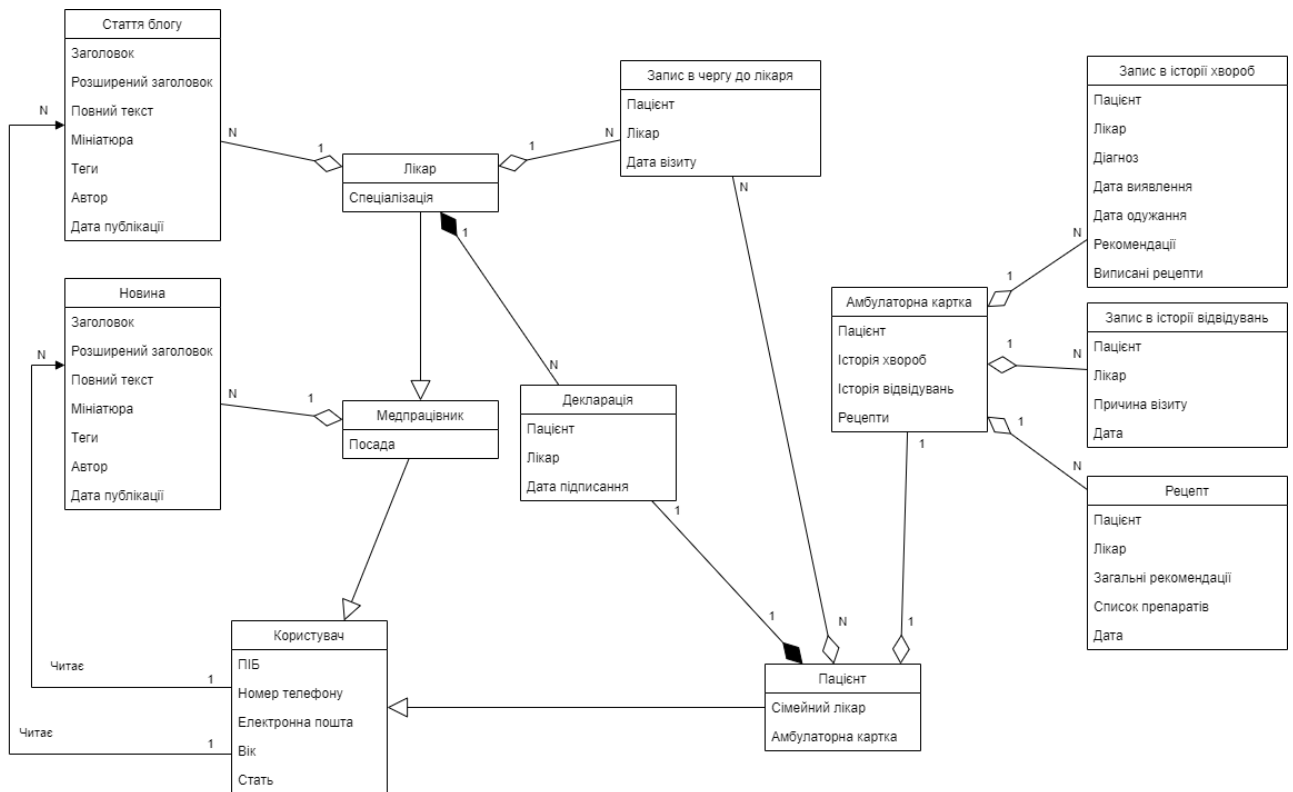


Рис. 3.1.1 UML діаграма концептуальних класів

### 3.1.2 Логічне проектування структур даних

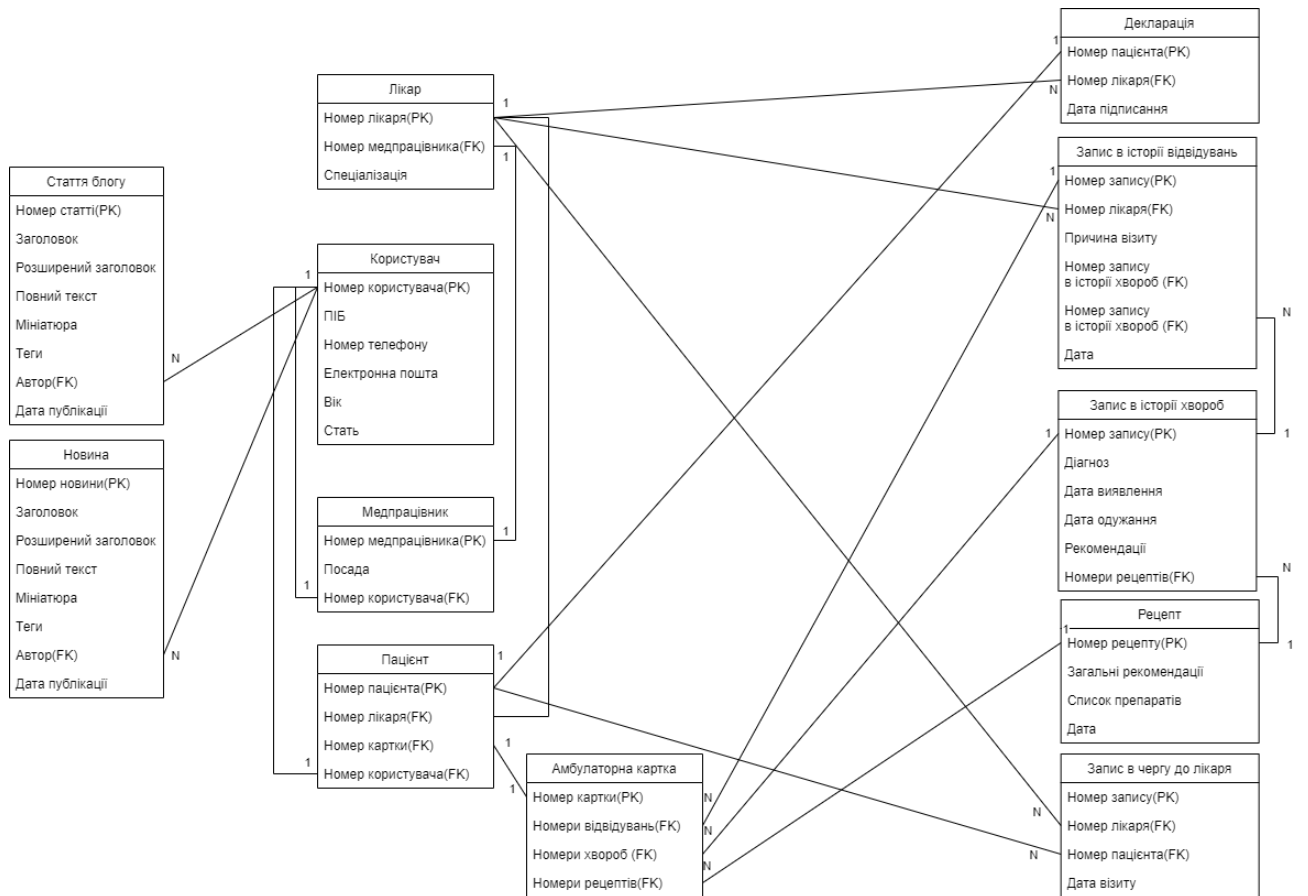


Рис. 3.1.2 Структура даних

## 3.2 Проектування програмних класів

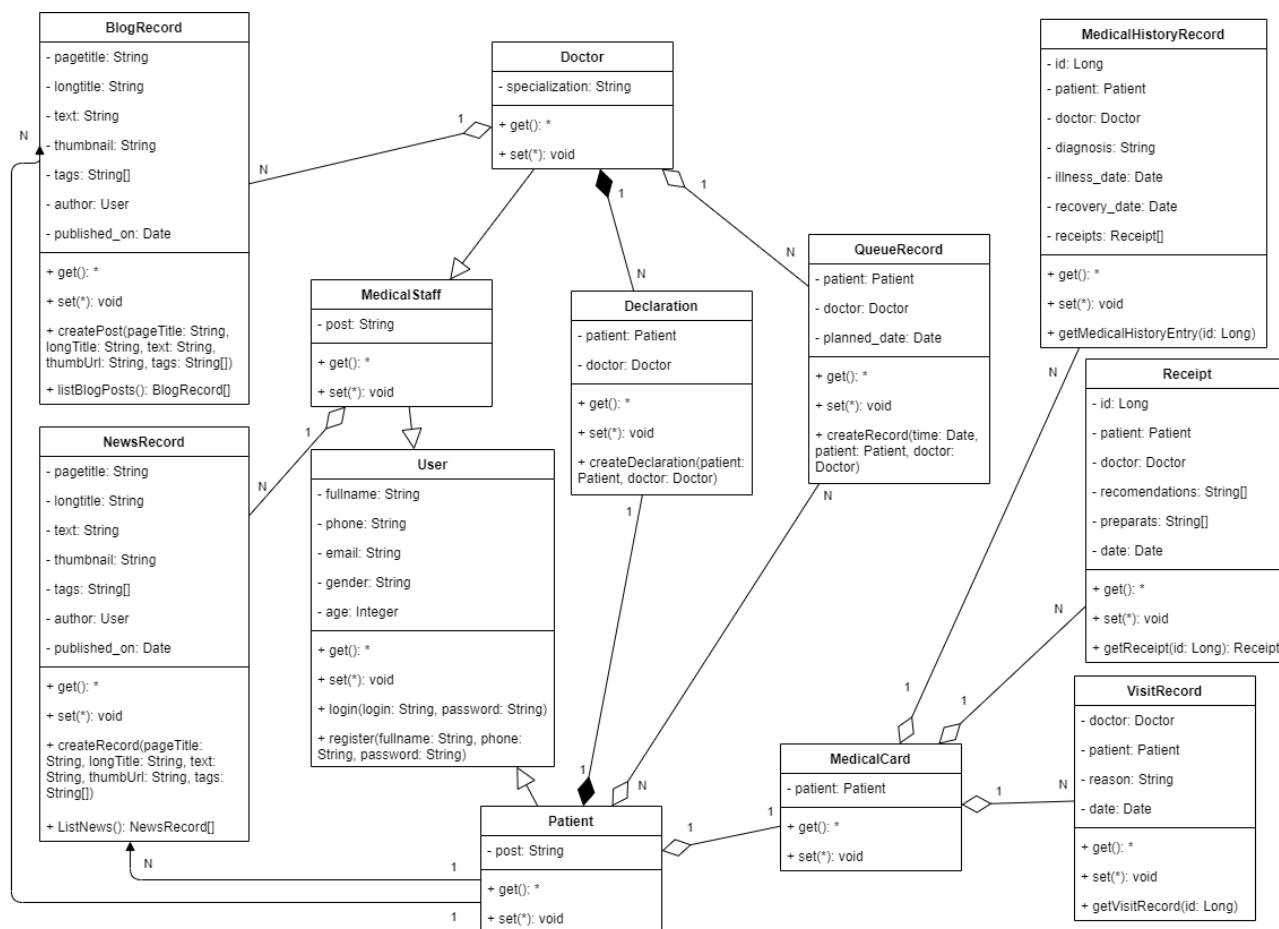


Рис. 3.1.2 UML діаграма програмних класів

## 3.3 Проектування алгоритмів роботи методів програмних класів

## 3.4 Проектування тестових сценаріїв верифікації роботи програмних модулів

### 3.4.1 Проектування тестових сценаріїв верифікації функціональних вимог

FR ID	Test ID	Опис значень вхідних даних до методу (процедури, функції)	Опис очікуваних значень результату виконання методу
FR 1.1	TC 1		
FR 1.2	TC 2		
FR 2.1	TC 3		
FR 2.2	TC 4		
FR 3.1	TC 5		

FR 3.1.1	TC 6		
FR 3.1.2	TC 7		
FR 3.1.3	TC 8		
FR 3.2	TC 9		
FR 3.3	TC 10		

### 3.4.2 Проектування тестових сценаріїв верифікації нефункціональних вимог

FR ID	Test ID	Опис значень вхідних даних до методу (процедури, функції)	Опис очікуваних значень результату виконання методу
FR 1	TC 1		
FR 2	TC 2		
FR 3	TC 3		
FR 4	TC 4		

## 4 Конструювання програмного продукту

### 4.1 Особливості конструювання структур даних

Framework Laravel надає можливість конструювати запити до бази даних за допомогою Eloquent ORM — засоба конструювання SQL запитів та методів керування базою даних мовою програмування, яка використовується на сервері.

Одним з методів керування БД — створення таблиць за допомогою класів міграцій. Кожен такий клас має розширювати клас Migration. Назва класу має містити ключові слова: Create, назва таблиці, Table. Зазвичай міграції створюються разом із моделлю за допомогою допоміжного скрипта artisan, який постачається з усіма проектами на базі фреймворка Laravel. Приклад створення моделі Receipt разом із класом міграції: *php artisan make:model Receipt -m*, де ключ *m* означає створення міграції. Її буде створено автоматично за допомогою імені моделі у множині, тобто отримаємо клас *CreateReceiptsTable*.

Класи міграцій містять два методи: *up*, *down*, у яких перший метод відповідає за створення структури даних, а другий за її коректне видалення або зпущення. Синтаксис опису поля структури даних наступний:

```
$table->min_даних("назва_поля")->обмеження_поля1()->обмеження2();
```

### 4.2 Особливості конструювання програмних модулів

Laravel — це фреймворк, який реалізує паттерн MVC. Тому уся бізнес-логіка прописується у контролерах. Створюються контролери за допомогою скрипта artisan наступною командою:

```
php artisan make:controller OrderController
```

Новий контролер буде створено у директорії *app/Http/Controllers*. Методами контролера дуже часто бувають *store()*, *index()*, *show()*, *destroy()*, *update()*, які разом мають відтворювати функціонал CRUD моделі.

Створення моделей також відбувається за допомогою скрипта artisan наступною командою:

```
php artisan make:model Order -mc,
```

де ключі *m*, *c* відповідають за створення міграцій та контролера відповідно. У класі моделі не прописуються її поля, їх записують у класах міграції. У моделі налаштовуються поля, які дозволено заповнювати, поля, які треба приховати. Також модель може містити вставки (traits), які відповідають за деякий додатковий функціонал. Наприклад, у даній курсовій роботі використано трейт *HasApiToken*, який дозволяє генерувати ключі авторизації. Також мають місце і методи відношень між сутностями, які повертають об'єкти прив'язаних

сутностей замість їхнього первинного ключа. Це стосується зв'язків «один-до-багатьох» та «багато-до-багатьох».

#### 4.2.1 Конструювання алгоритмів методів програмних класів або процедур/функцій

Далі наведено код усіх контролерів:

*NewsController:*

```
<?php

namespace App\Http\Controllers;

use App\Models\NewsRecord;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class NewsController extends Controller
{
    public function index()
    {
        $news = NewsRecord::all();
        return $news;
    }

    public function store(Request $request)
    {
        if (Auth::user()->role_id == 4) {
            return response(null, 403);
        }

        $record = NewsRecord::create([
            'title' => $request->title,
            'longtitle' => $request->longtitle,
            'text' => $request->text,
            'user_id' => $request->author
        ]);

        return response($record, 201);
    }

    public function show(NewsRecord $record)
    {
        return response($record);
    }

    public function destroy(NewsRecord $record)
    {
        if (Auth::user()->role_id == 4) {
```

```

        return response(null, 403);
    }

    if ($record->delete()) {
        return response(null, 204);
    }
}

public function update(Request $request, NewsRecord
$record)
{
    if (Auth::user()->role_id == 4) {
        return response(null, 403);
    }

    $post = $record->update([
        'title' => $request->title,
        'longtitle' => $request->longtitle,
        'text' => $request->text
    ]);

    return response($record, 201);
}
}

```

### *VisitController:*

```

<?php

namespace App\Http\Controllers;

use App\Models\VisitRecord;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class VisitController extends Controller
{
    public function index($patient)
    {
        if (Auth::user()->role_id != 4 || Auth::id() ==
$patient) {
            return VisitRecord::all()->where('patient_id',
$patient);
        }

        return response(null, 403);
    }

    public function store($patient, Request $request)
    {
        if (Auth::user()->role_id != 3) {
            return response(null, 403);
        }
    }
}

```



```

    }

    $record = VisitRecord::create([
        'doctor_id' => Auth::id(),
        'patient_id' => intval($patient),
        'reason' => $request->reason
    ]);

    return response($record, 201);
}

public function show(VisitRecord $visit)
{
    $id = Auth::id();

    if ($id == $visit->doctor_id || $id == $visit->patient_id || Auth::user()->role_id == 1 || Auth::user()->role_id == 2) {
        return $visit;
    }

    return response(null, $id ? 403 : 401);
}

public function destroy(VisitRecord $visit)
{
    if (Auth::user()->role_id == 1 || Auth::id() == $visit->doctor_id) {
        if ($visit->delete()) {
            return response(null, 204);
        }
    }
    return response(null, 403);
}
}
}

```

### *StaffController:*

```

<?php

namespace App\Http\Controllers;

use App\Models\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Hash;

class StaffController extends Controller
{
    public function index()
    {

```

```

        if (Auth::user()->role_id == 4) {
            return response(null, 403);
        }

        return response()->json(User::whereIn('role_id', [2,
3]))->get());
    }

    public function store(Request $request)
    {
        if (Auth::user()->role_id == 4) {
            return response(null, 403);
        }

        $input = $request->all();
        if (isset($input['password'])) {
            $input['password'] =
Hash::make($input['password']);
        }

        if (isset($input['doctor'])) {
            $input['role_id'] = 3;
            unset($input['doctor']);
        } else {
            $input['role_id'] = 2;
        }

        $user = User::create($input);

        return response($user, 201);
    }

    public function destroy(User $user)
    {
        if (Auth::user()->role_id == 4) {
            return response(null, 403);
        }

        if ($user->delete()) {
            return response(null, 204);
        }
    }
}

```

### ***AuthController:***

```

<?php

namespace App\Http\Controllers\API;

use App\Http\Controllers\Controller;
use App\Models\User;

```

```

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Facades\Validator;

class AuthController extends Controller
{
    public function register(Request $request)
    {
        $validator = Validator::make($request->all(), [
            'name' => ['required', 'string', 'max:255'],
            'email' => ['required', 'string', 'email',
'max:255', 'unique:users'],
            'password' => ['required', 'string', 'min:8']
        ]);

        if ($validator->fails()) {
            return response()->json(['error' => $validator-
>errors()], 401);
        }

        $input = $request->all();
        $input['password'] = bcrypt($input['password']);
        $input['role_id'] = 4;

        $user = User::create($input);

        $token      =      $user->createToken($request->email)-
>plainTextToken;

        return response()->json(['token' => $token], 200);
    }

    public function token(Request $request)
    {
        $validator = Validator::make($request->all(), [
            'email' => ['required', 'string', 'email',
'max:255'],
            'password' => ['required', 'string', 'min:8']
        ]);

        if ($validator->fails()) {
            return response()->json(['error' => $validator-
>errors()], 401);
        }

        $user      =      User::where('email',      $request->email)-
>first();

        if (!$user || !Hash::check($request->password, $user-
>password)) {
            return response()->json(['error' => 'The provided
credentials are incorrect: '], 401);
        }
    }
}

```

```

        }

        return response()->json(['token' => $user-
>createToken($request->email)->plainTextToken]);
    }
}

```

## 4.2 Модульне тестування програмних модулів

Для модульного тестування застосовано бібліотеку PHPUnit. Код Unit тестів:

```

class AuthTest extends TestCase
{
    public function test_login_admin()
    {
        $response = $this->post('/api/sanctum/token', ['email' =>
'admin@comed', 'password' => 'admin123']);

        $response->assertStatus(200);
    }

    public function test_login_incorrect()
    {
        $response = $this->post('/api/sanctum/token', ['email' =>
'tspp@comed', 'password' => '880005']);

        $response->assertStatus(401);
    }

    public function test_register()
    {
        $response = $this->post('/api/sanctum/register', ['email'
=> 'tspp20214@comed', 'name' => 'Тестовий персонаж', 'password' =>
'880005567']);
        $response->assertStatus(200);
    }

    public function test_register_fail()
    {
        $response = $this->post('/api/sanctum/register', ['email'
=> 'tspp2021@comed', 'password' => '880005']);
        $response->assertStatus(401);
    }
}

```

Процес модульного тестування запустимо командою *php artisan test*:

```
PS C:\Users\yurdm\projects\comed-backend> php artisan test
Warning: TTY mode is not supported on Windows platform.

PASS Tests\Unit\AuthTest
✓ login admin
✓ login incorrect
✓ register
✓ register fail

Tests: 4 passed
Time: 0.44s

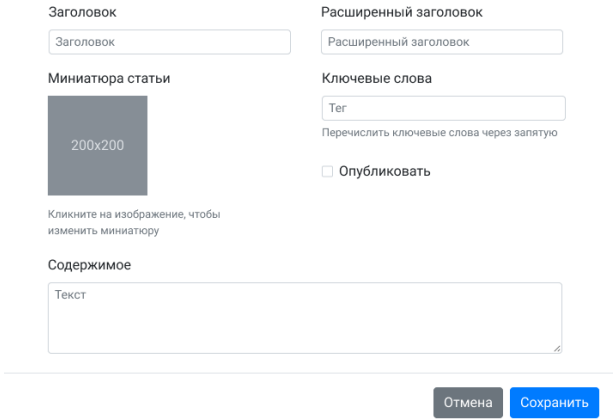
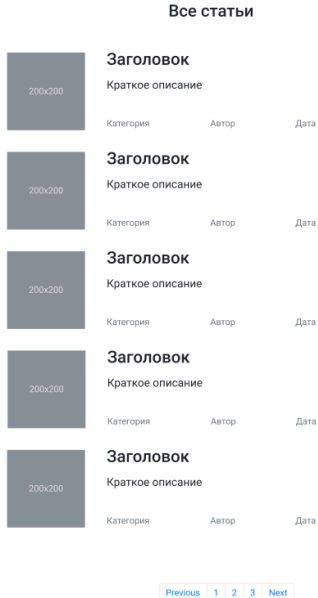
PS C:\Users\yurdm\projects\comed-backend> 
```

Рис. 4.2.1 Тестування контролера авторизації




## 5 Верифікація програмного продукту

### 5.1 Тестування апаратно-програмних інтерфейсів програмного продукту

#### 5.2 Тестування інтерфейсу користувача програмного продукту

NFR ID	Test Case ID	Опис дій з ПП на рівні інтерфейсу користувача	Опис очікуваних результатів взаємодії з ПП на рівні інтерфейсу користувача
FNR 1	TC 1	Перейти на Адмін панель та натиснути кнопку «Створити блог»	
FNR 2	TC 2	Натиснути на кнопку «Блоги»	

FNR 3	TC 3	Перейти на Адмін панель та натиснути кнопку «Створити новину»	<div> <div>Заголовок</div> <div>Заголовок</div> </div> <div> <div>Расширенный заголовок</div> <div>Расширенный заголовок</div> </div> <div> <div>Миниатюра статьи</div> <div>200x200</div> <div>Кликните на изображение, чтобы изменить миниатюру</div> </div> <div> <div>Ключевые слова</div> <div>Тег</div> <div>Перечислите ключевые слова через запятую</div> <div><input type="checkbox"/> Опубликовать</div> </div> <div> <div>Содержимое</div> <div>Текст</div> </div> <div> <div>Отмена</div> <div>Сохранить</div> </div>
FNR 4	TC 4	Натиснути на кнопку «Новини»	<div>Все статьи</div> <div> <div>200x200</div> <div>Заголовок</div> <div>Краткое описание</div> <div>Категория</div> <div>Автор</div> <div>Дата</div> </div> <div> <div>200x200</div> <div>Заголовок</div> <div>Краткое описание</div> <div>Категория</div> <div>Автор</div> <div>Дата</div> </div> <div> <div>200x200</div> <div>Заголовок</div> <div>Краткое описание</div> <div>Категория</div> <div>Автор</div> <div>Дата</div> </div> <div> <div>200x200</div> <div>Заголовок</div> <div>Краткое описание</div> <div>Категория</div> <div>Автор</div> <div>Дата</div> </div> <div> <div>200x200</div> <div>Заголовок</div> <div>Краткое описание</div> <div>Категория</div> <div>Автор</div> <div>Дата</div> </div> <div> <div>Previous</div> <div>1</div> <div>2</div> <div>3</div> <div>Next</div> </div>
FNR 5	TC 5	Натиснути кнопку «Особистий кабінет»	<div>Личный кабинет</div> <div> <div> <div>Карта пациента</div> <div>Перейти в раздел</div> </div> <div> <div>Запись к врачу</div> <div>Выбрать врача</div> </div> <div> <div>Семейный врач</div> <div>Перейти в раздел</div> </div> </div>

FNR 6	TC 6	Натиснувши кнопку «Картка пацієнта» у особистому кабінеті	<p><b>Амбулаторна картка</b></p> <p>ПІБ: _____ Сімейний лікар: _____</p> <p>Номер телефону: _____</p> <p>Вік: _____</p> <p><a href="#">Редагувати мої дані</a></p> <div> <div>  <p>Історія відвідувань</p> <p><a href="#">Перейти</a></p> </div> <div>  <p>Історія хвороб</p> <p><a href="#">Перейти</a></p> </div> <div>  <p>Мої рецепти</p> <p><a href="#">Перейти</a></p> </div> </div>																				
FNR 7	TC 7	Натиснувши кнопку «Історія хвороб» на сторінці картки пацієнта	<p><b>История болезней</b></p> <table> <tr> <th>Диагноз</th><th>Дата установления диагноза</th><th>Дата выздоровления</th><th></th></tr> <tr> <td>ОРВИ</td><td>15.06.2021</td><td>18.06.2021</td><td><a href="#">Подробнее</a></td></tr> <tr> <td>ОРВИ</td><td>15.06.2021</td><td>18.06.2021</td><td><a href="#">Подробнее</a></td></tr> <tr> <td>ОРВИ</td><td>15.06.2021</td><td>18.06.2021</td><td><a href="#">Подробнее</a></td></tr> </table>	Диагноз	Дата установления диагноза	Дата выздоровления		ОРВИ	15.06.2021	18.06.2021	<a href="#">Подробнее</a>	ОРВИ	15.06.2021	18.06.2021	<a href="#">Подробнее</a>	ОРВИ	15.06.2021	18.06.2021	<a href="#">Подробнее</a>				
Диагноз	Дата установления диагноза	Дата выздоровления																					
ОРВИ	15.06.2021	18.06.2021	<a href="#">Подробнее</a>																				
ОРВИ	15.06.2021	18.06.2021	<a href="#">Подробнее</a>																				
ОРВИ	15.06.2021	18.06.2021	<a href="#">Подробнее</a>																				
FNR 8	TC 8	Натиснувши кнопку «Історія відвідувань» на сторінці картки пацієнта	<p><b>История посещений</b></p> <table> <tr> <th>Врач</th><th>Время визита</th><th>Цель визита</th><th></th></tr> <tr> <td>Пилиюлькин А.В.</td><td>15.06.2021 15:30</td><td>Жалобы на кашель</td><td><a href="#">Подробнее</a></td></tr> <tr> <td>Пилиюлькин А.В.</td><td>15.06.2021 15:30</td><td>Жалобы на кашель</td><td><a href="#">Подробнее</a></td></tr> <tr> <td>Пилиюлькин А.В.</td><td>15.06.2021 15:30</td><td>Жалобы на кашель</td><td><a href="#">Подробнее</a></td></tr> <tr> <td>Пилиюлькин А.В.</td><td>15.06.2021 15:30</td><td>Жалобы на кашель</td><td><a href="#">Подробнее</a></td></tr> </table>	Врач	Время визита	Цель визита		Пилиюлькин А.В.	15.06.2021 15:30	Жалобы на кашель	<a href="#">Подробнее</a>	Пилиюлькин А.В.	15.06.2021 15:30	Жалобы на кашель	<a href="#">Подробнее</a>	Пилиюлькин А.В.	15.06.2021 15:30	Жалобы на кашель	<a href="#">Подробнее</a>	Пилиюлькин А.В.	15.06.2021 15:30	Жалобы на кашель	<a href="#">Подробнее</a>
Врач	Время визита	Цель визита																					
Пилиюлькин А.В.	15.06.2021 15:30	Жалобы на кашель	<a href="#">Подробнее</a>																				
Пилиюлькин А.В.	15.06.2021 15:30	Жалобы на кашель	<a href="#">Подробнее</a>																				
Пилиюлькин А.В.	15.06.2021 15:30	Жалобы на кашель	<a href="#">Подробнее</a>																				
Пилиюлькин А.В.	15.06.2021 15:30	Жалобы на кашель	<a href="#">Подробнее</a>																				
FNR 9	TC 9	Натиснувши кнопку «Список рецептов» на сторінці картки пацієнта	<p><b>Список рецептов</b></p> <table> <tr> <th>Идентификатор</th><th>Дата приписания</th><th>Кто выписал</th><th></th></tr> <tr> <td>№12345678</td><td>15.06.2021</td><td>Пилиюлькин А.В.</td><td><a href="#">Подробнее</a></td></tr> <tr> <td>№12345678</td><td>15.06.2021</td><td>Пилиюлькин А.В.</td><td><a href="#">Подробнее</a></td></tr> <tr> <td>№12345678</td><td>15.06.2021</td><td>Пилиюлькин А.В.</td><td><a href="#">Подробнее</a></td></tr> <tr> <td>№12345678</td><td>15.06.2021</td><td>Пилиюлькин А.В.</td><td><a href="#">Подробнее</a></td></tr> </table>	Идентификатор	Дата приписания	Кто выписал		№12345678	15.06.2021	Пилиюлькин А.В.	<a href="#">Подробнее</a>	№12345678	15.06.2021	Пилиюлькин А.В.	<a href="#">Подробнее</a>	№12345678	15.06.2021	Пилиюлькин А.В.	<a href="#">Подробнее</a>	№12345678	15.06.2021	Пилиюлькин А.В.	<a href="#">Подробнее</a>
Идентификатор	Дата приписания	Кто выписал																					
№12345678	15.06.2021	Пилиюлькин А.В.	<a href="#">Подробнее</a>																				
№12345678	15.06.2021	Пилиюлькин А.В.	<a href="#">Подробнее</a>																				
№12345678	15.06.2021	Пилиюлькин А.В.	<a href="#">Подробнее</a>																				
№12345678	15.06.2021	Пилиюлькин А.В.	<a href="#">Подробнее</a>																				
FNR 10	TC 10	Натиснувши кнопку «Запис до лікаря» у особистому кабінеті	<p><b>Запись на приём</b></p> <p>Введите ФИО врача либо его специализацию <a href="#">Поиск</a></p> <table> <tr> <td></td> <td> <b>Пилиюлькин А.В.</b>  Педиатр </td> <td> Пн-Сб  10:00 – 21:00 </td> <td><a href="#">Выбрать время</a></td> </tr> <tr> <td></td> <td> <b>Пилиюлькин А.В.</b>  Педиатр </td> <td> Пн-Сб  10:00 – 21:00 </td> <td><a href="#">Выбрать время</a></td> </tr> <tr> <td></td> <td> <b>Пилиюлькин А.В.</b>  Педиатр </td> <td> Пн-Сб  10:00 – 21:00 </td> <td><a href="#">Выбрать время</a></td> </tr> </table>		<b>Пилиюлькин А.В.</b> Педиатр	Пн-Сб 10:00 – 21:00	<a href="#">Выбрать время</a>		<b>Пилиюлькин А.В.</b> Педиатр	Пн-Сб 10:00 – 21:00	<a href="#">Выбрать время</a>		<b>Пилиюлькин А.В.</b> Педиатр	Пн-Сб 10:00 – 21:00	<a href="#">Выбрать время</a>								
	<b>Пилиюлькин А.В.</b> Педиатр	Пн-Сб 10:00 – 21:00	<a href="#">Выбрать время</a>																				
	<b>Пилиюлькин А.В.</b> Педиатр	Пн-Сб 10:00 – 21:00	<a href="#">Выбрать время</a>																				
	<b>Пилиюлькин А.В.</b> Педиатр	Пн-Сб 10:00 – 21:00	<a href="#">Выбрать время</a>																				



FNR 11	ТС 11	Натиснувши кнопку «Сімейний лікар» у особистому кабінеті	<div> <div>Выбор семейного врача</div> <div> <div>Введите ФИО врача</div> <div>Поиск</div> </div> <div> <div> <div>200x200</div> <div> <div>Пилюлькин А.В.</div> <div>Педиатр</div> </div> <div> <div>Пн-Сб</div> <div>10:00 – 21:00</div> </div> <div>Подать заявку</div> </div> <div> <div>200x200</div> <div> <div>Пилюлькин А.В.</div> <div>Педиатр</div> </div> <div> <div>Пн-Сб</div> <div>10:00 – 21:00</div> </div> <div>Подать заявку</div> </div> <div> <div>200x200</div> <div> <div>Пилюлькин А.В.</div> <div>Педиатр</div> </div> <div> <div>Пн-Сб</div> <div>10:00 – 21:00</div> </div> <div>Подать заявку</div> </div> </div> </div>
FNR 12	ТС 12	Натиснувши кнопку «Вхід»	<div> <div>Авторизация</div> <div> <div>Введите ваш Email...</div> <div>Введите ваш пароль...</div> <div> <div>Нет аккаунта? <a href="#">Зарегистрируйтесь!</a></div> <div>Войти</div> </div> </div> </div>

## 6 Розгортання та валідація програмного продукту

### 6.1 Інструкція з встановлення системного програмного забезпечення

За рисунком 2.1 ми бачимо, що система інсталується на трьох серверах: один для БД (Access level), один для backend частини (Business level), і один для frontend частини (Presentation level).

Для даної курсової роботи інсталяція проводилася на сервісах Heroku та Google Firebase Hosting. У репозиторії програмного продукту вказано посилання на репозиторії для frontend та backend частин. Репозиторії клонуються на окремі сервери. В нашому випадку інсталяція backend проводилася на Heroku, а інсталяція frontend — на Firebase.

Після клонування репозиторія серверної частини продукту перш за все слід завантажити додаткові пакети командою *composer install*. Слід звернути увагу, що на сервері інстальовано інтерпретатор для мови програмування PHP версії 7.3 як мінімум, а також інстальовано пакетний менеджер *composer*. Для роботи веб сервера використовувати Apache HTTP Server.

Далі слід скопіювати файл *.env.example* і вставити у той же каталог як *.env*. Це файл конфігурації застосунку. Потім у командному рядку згенерувати ключ проекту: *php artisan key:generate*. У файлі *.env* у опції, що починаються на DB, ввести дані підключення бази даних, яку теж слід створити окремо. Після успішного підключення бази даних слід запустити міграції для ініціалізації БД за допомогою команди *php artisan make:migrate --seed*.

Для інсталяції клієнтської частини на сервер потрібно клонувати репозиторій із frontend частиною. Попередньо на сервер інстальувати *nodejs* та пакетний менеджер *npm*. Для початку слід інстальувати усі модулі командою *npm install*, а потім зібрати проект командою *npm run build*. Зібраний проект з'явиться у директорії *build*. Для початку роботи треба налаштувати HTTP сервер (NGINX, Apache, на вибір): усі запити до сервера мають бути спрямовані на *index.html*, так як це SPA застосунок.

## 6.2 Інструкція з використання програмного продукту

Користувач переходить за посиланням та потрапляє на сайт мед. закладу, де він може виконувати різні дії, а саме:

- a. Переглядати новини закладу;
- b. Переглядати список рецептів;
- c. Переглядати амбулаторну картку;
- d. Переглядати історію хвороб та ін.

Якщо у користувача є допуск медперсоналу, він також має доступ до частини функцій адмін панелі: створювати новини, заповнювати картку пацієнта

## 6.3 Результати валідації програмного продукту

Кроки, що підтверджують досягнення цілі ПП:

- 1. користувач реєструється на сайті;
- 2. користувач переходить до особистого кабінету;
- 3. користувач записується на відвідування лікаря;
- 4. користувач переглядає виписані для нього рецепти;

Кількість елементарних операцій, які користувач виконував раніше, до використання ПП:

- 1. похід до лікарні
- 2. заведення особистої картки
- 3. запис до черги лікаря на наступний день
- 4. візит до лікарні
- 5. перегляд виписаних рецептів

Отже, користувач виконував 5 кроків для досягнення мети.

## Висновки

У цій роботі спроектовано веб застосунок системи обміну корпоративними даними у медичному закладі. Його практичне застосування полягає у забезпеченні доступу до особистої інформації для пацієнтів та зручного обміну даними всередині медичного закладу.

В процесі створення програмного продукту виникли такі труднощі (організаційні, проблеми відсутності досвіду, знань, потрібних в різних етапах):

1. розгортання продукту на сервер;
2. налаштування CORS для різних доменів;
3. організаційні проблеми;
4. проблеми із розробкою frontend частини через відсутність достатньої кількості знань та досвіду

Через вищеописані непередбачені труднощі, а також через обмежений час на створення програмного продукту, залишаються нереалізованими такі прецеденти або їх окремі кроки роботи:

1. підписання декларацій;
2. запис на прийом
3. електронні рецепти
4. блоги

Зазначені недоробки планується реалізувати в майбутніх курсових роботах з урахуванням тем дисциплін наступних семестрів.