Міністерство освіти і науки України

Луцький національний технічний університет

Факультет комп'ютерних та інформаційних технологій

Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

СТВОРЕННЯ КЛІЄНТ-СЕРВЕРНОГО ДОДАТКУ ДЛЯ ОРГАНІЗАЦІЇ ЕЛЕКТРОННОЇ ЧЕРГИ З ВИКОРИСТАННЯМ DJANGO

CREATING A CLIENT-SERVER APPLICATION FOR ORGANIZING AN ELECTRONIC QUEUE USING DJANGO

спеціальність 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

освітня програма «Інженерія програмного забезпечення»
(назва освітньої програми)

Виконав: здобувач вищої освіти
Групи ІПЗс-21
Бортнюк Андрій Юрійович
_____
(підпис)

Керівник:
к.т.н., доцент
Суринович Олена Миколаївна
_____
(підпис)

Кваліфікаційну роботу
допущено до захисту
«8» 06 2024 р.
к.т.н., доцент
Гарант освітньої програми:
Ліщина Наталія Миколаївна
_____
(підпис)

Луцьк – 2024 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *інженерії програмного забезпечення*
Ступінь вищої освіти: *бакалавр*
Галузь знань: *12 Інформаційні технології*
Спеціальність: *121 Інженерія програмного забезпечення*
Освітня програма: *«Інженерія програмного забезпечення»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ *Н. Кіщук*
«*2*» *02* 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Бортнюку Андрію Юрійовичу
(прізвище, ім`я, по батькові)

1. Тема кваліфікаційної роботи: *«Створення клієнт-серверного додатку для організації електронної черги з використанням Django».*

Керівник роботи: *к.т.н., доцент, Суринович Олена Миколаївна*
затверджені наказом закладу вищої освіти від «*30» грудня 2023 р.* № *477/01-02*
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи *«5» червня 2024 р.*
3. Вихідні дані до роботи: *технічне та програмне забезпечення ЕОМ, вимоги до розробки програмного забезпечення, ергономічні вимоги до функціонування програмного засобу.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити): *Аналіз сучасного стану проблеми, існуючих методів і засобів її розв'язання, аналіз, визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення, опис функціонального наповнення об'єкта проектування, практична реалізація об'єкта проектування.*

5. Перелік графічного матеріалу: *15 рисунків, з яких 2 діаграми та 2 схеми.*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз предметної області	*Суринович О. М.*		
Специфікації вимог до розробленої системи	*Суринович О. М.*		
Розробка об'єкта проєктування	*Суринович О. М.*		
Нормоконтроль	*Повстяна Ю. С.*		
Гарант ОП	*Ліщина Н. М.*		
Показник запозичень тексту	2,8 %		
Академічна доброчесність	*Ящук А. А.*		

7. Дата видачі завдання *«2» лютого 2024 р.*

КАЛЕНДАРНИЙ ПЛАН

№ З/п	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1	*Огляд літературних джерел по темі кваліфікаційної роботи бакалавра*	*07.02.2024 р.*	вик.
2	*Аналіз проблеми розробки та впровадження об'єкту проектування*	*27.02.2024 р.*	вик.
3	*Обґрунтування вибору шляхів, технологій (алгоритмів) і засобів вирішення поставленого завдання*	*12.03.2024 р.*	вик.
4	*Розробка функціонально-структурної схеми роботи об'єкта проектування та проектування бази даних*	*17.04.2024 р.*	вик.
5	*Практична реалізація об'єкта проектування та розробка бази даних*	*18.05.2024 р.*	вик.
6	*Тестування та налагодження об'єкта проектування*	*01.06.2024 р.*	вик.
7	*Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедру*	*05.06.2024 р.*	вик.

Здобувач вищої освіти _____ (підпис) *Бортнюк А. Ю.* (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ (підпис) *Суринович О. М.* (прізвище, ініціали)

Міністерство освіти і науки України
Луцький національний технічний університет

Рецензія
на кваліфікаційну роботу

Здобувач вищої освіти: *Бортнюк Андрій Юрійович*
Тема: *«Створення клієнт-серверного додатку для організації електронної черги з використанням Django».*

Коротка характеристика кваліфікаційної роботи:

Кваліфікаційна робота бакалавра складається з трьох розділів, в яких проведено аналіз предметної області, здійснена чітка постановка завдань за темою роботи, проведено дослідження методів та засобів для розробки клієнт-серверного додатку, здійснено практичну реалізацію і проведено дослідження результату ефективності додатку, а також здійснено аналіз отриманих результатів.

Самостійні розробки і пропозицій автора:

Автором було самостійно розроблено клієнт-серверний додаток для організації електронної черги з використанням Django

Практичне значення роботи

полягає в ефективному використанні розробленого додатку для будь-яких організацій, що використовують електронну чергу.

Недоліки: *Суттєвих недоліків в роботі не виявлено.*

Загальний висновок

Кваліфікаційна робота бакалавра виконана на достатньому рівні. Робота виконана з розумінням предмету. Матеріал викладено в чіткій та зручній формі. Вважаю, що кваліфікаційна робота відповідає вимогам до випускних робіт і заслуговує позитивної оцінки, а її авторові, студенту Бортнюку А. Ю. може бути присвоєний кваліфікаційний рівень "бакалавр" зі спеціальності «Інженерія програмного забезпечення»

Рецензент: *Заблоцький В. Ю., к. т. н., доц., завідувач кафедри електроніки та телекомунікацій*

Рецензент кваліфікаційної роботи _____
(підпис)

«7» червня 2024 р.

Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення

ВІДГУК
КЕРІВНИКА НА КВАЛІФІКАЦІЙНУ РОБОТУ

Здобувач вищої освіти *Бортнюк Андрій Юрійович*
(прізвище, ім'я, по батькові)
група *ІПЗс-21*

Тема кваліфікаційної роботи: *«Створення клієнт-серверного додатку для організації електронної черги з використанням Django».*

Керівник *Суринович Олена Миколаївна*

Актуальність теми *Пандемія коронавірусу COVID-19, яка за п'ять років практично охопила всі континенти і зачепила 90% країн світу, є випробуванням для людства на здатність протистояти загальноцивілізаційним загрозам, організовуватися для вирішення важливих проблем і робити правильні висновки. Тому для убезпечення людей на майбутнє постає питання розробки додатку з управління чергами у закладах охорони здоров'я.*

Об'єкт дослідження *Клієнт-серверний додаток для організації електронної черги з використанням Django.*

Характеристика теоретичного рівня, наявності самостійних розробок і практичної значимості роботи: *Автор аналізує різні методи та засоби для розробки клієнт-серверного додатку для організації електронної черги з використанням Django. Визначає найкращі з них і обґрунтовує своє рішення. Зміст роботи відповідає обраній темі. Позитивними рисами кваліфікаційної роботи бакалавра є системність та послідовність викладення матеріалу, а також якісна його практична реалізація.*

Зауваження та недоліки: *Суттєвих недоліків в роботі не виявлено*

Загальний висновок: *Кваліфікаційна робота бакалавра виконана на високому рівні. Робота виконана відповідно до вимог, логічно й послідовно описує хід виконання поставленого завдання. Пояснювальна записка відповідає стандартам до її оформлення. Дана робота заслуговує позитивної оцінки.*

Керівник _____ **(*Суринович О. М*)**
(підпис) (прізвище, ім'я, по батькові)

«7» червня 2024 р.

АНОТАЦІЯ

Бортнюк А. Ю. Створення клієнт-серверного додатку для організації електронної черги з використанням Django. Рукопис.

Кваліфікаційна робота бакалавра ОП «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2024.

Кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, висновків та списку використаних джерел.

У першому розділі здійснено аналіз предметної області і постановка завдань на кваліфікаційну роботу. В другому розділі визначено специфікацію вимог до розроблюваної системи. У третьому розділі описано етапи розробки клієнт-серверного додатку для організації електронної черги. У висновках узагальнено інформацію, відображену у попередніх частинах. Результати розробки демонструють можливості розробленого додатку.

Ключові слова: клієнт-серверний додаток, Django, соціальне дистанціювання, керування потоками відвідувачів, організація контролю та обліку співробітників.

ABSTRACT

Bortniuk A. Creating a Client-Server Application for Organizing an Electronic Queue Using Django. Manuscript.

Bachelor's qualifying thesis of the educational program "Software Engineering". Lutsk National Technical University. Lutsk, 2024.

The bachelor’s qualifying thesis consists of an introduction, three sections, conclusions and references.

In the first section, the analysis of the subject area and the setting of tasks for the qualification work were carried out. The second section defines the specification of requirements for the developed system. The third section describes the stages of development of a client-server application for organizing an electronic queue. The conclusions summarize the information presented in the previous parts. The development results demonstrate the capabilities of the developed application.

Keywords: client-server application, Django, social distancing, management of visitor flows, organization of control and accounting of employees.

ЗМІСТ

ВСТУП

Актуальність роботи. Пандемія коронавірусу COVID-19, яка за п’ять років практично охопила всі континенти і зачепила 90 % країн світу, є випробуванням для людства на здатність протистояти загальноцивілізаційним загрозам, організовуватися для вирішення важливих проблем і робити правильні висновки.

Головною метою кваліфікаційної роботи бакалавра є розробка клієнт-серверного додатку для управління електронною чергою до лікаря. Розробка такого додатка дозволить вирішити проблему організації соціального дистанціювання у місцях підвищеної небезпеки поширення небезпечних захворювань, таких як COVID-19, за рахунок зменшення ймовірностей можливих фізичних контактів, заражених з неінфікованими людьми. Також система допоможе формалізувати та оптимізувати управління потоками відвідувачів, допомогти підвищити якість обслуговування відвідувачів лікарень шляхом автоматичного розподілу потоків відвідувачів та організації контролю та обліку співробітників, які здійснюють обслуговування.

Для реалізації мети слід виконати наступні завдання:

– проаналізувати предметну область;

– провести дослідження існуючих аналогів;

– навчитися працювати з Python фреймворк Django;

– навчитися працювати з Django Rest Framework;

– реалізувати систему авторизації та реєстрації;

– реалізувати моделі «Лікаря», «Пацієнта» та «Адміністратора системи»;

– для керування системою реалізувати клієнт-серверний додаток;

– реалізувати API сервісу;

– навчитися працювати з фреймворком UIKit;

– реалізувати можливість потрапляння в чергу пацієнтів за допомогою зчитування QR коду користувачем мобільного додатку;

– реалізувати механізм отримання даних додатком.

Об’єкт кваліфікаційної роботи бакалавра – клієнт-серверний додаток для організації електронної черги з використанням Django.

Предмет кваліфікаційної роботи бакалавра – засоби розробки додатку для організації електронної черги.

Практична цінність кваліфікаційної роботи бакалавра полягає у використанні програмного продукту пацієнтами та працівниками закладу охорони здоров’я для мінімізації ризиків зараження вірусною інфекцією та оптимізації управління потоками відвідувачів.

РОЗДІЛ 1
АНАЛІЗ КЛІЄНТ-СЕРВЕРНОЇ АРХІТЕКТУРИ І ПОСТАНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ

1.1 Аналіз сучасного стану проблеми

П’ять років тому людство зіштовхнулося із масштабною проблемою, вірусом COVID-19, до якого воно було зовсім не підготовлено. Згідно зі статистикою, що надається John Hopkins University – до квітня 2021 року кількість інфікованих небезпечним вірусом перевищило сто мільйонів людей, включаючи 2,3 мільйони смертей. На початку пандемії уряд, наукові центри, ВООЗ не мали чіткого розуміння та способу вирішення небезпечної проблеми. Але стало зрозуміло відразу, що найкращим комплексом заходів, спрямованих на запобігання поширенню коронавірусу є – соціальне дистанціювання та обмеження фізичного контакту між людьми.

За статистикою ВООЗ, найбільша ймовірність заразитися COVID-19 зберігається в місцях скупчення людей, у замкнутих просторах з поганою вентиляцією, зокрема, у лікарнях та поліклініках.

Протягом квітня 2024 року у США, Великій Британії та Канаді швидко поширюється новий штам коронавірусу FLiRT, який є похідним від субваріанту «Омікрон – JN.1», також відомого як «Дженні». Найбільш відомим серед штамів FLiRT є варіант KP.2.

Про це повідомили фахівці Школи громадської охорони здоров’я Університету Джонса Гопкінса у США [1].

Тому для убезпечення людей на майбутнє постає питання розробки додатку з управління чергами у закладах охорони здоров’я. Варто зазначити, що така система надасть безліч плюсів, як організації, так її клієнтам. Для відвідувачів процедура очікування своєї черги стане зрозумілою та прогнозованою, що, у свою чергу, знизить рівень їхньої нервозності та убезпечить від контакту з хворою людиною. Для організації – відкриває нові можливості при ухваленні рішень.

Розглянемо два аналоги в Україні, що містять функцію попереднього запису до лікаря. Однією з найпопулярніших на даний час систем є застосунок «Helsi» – зручна медична інформаційна система для пацієнтів, яка дозволяє швидко знаходити лікаря онлайн у будь-якому куточку України, а також записувати себе чи своїх рідних до лікаря на зручний день і час на відео-прийом або в медичний заклад (рис. 1.1).

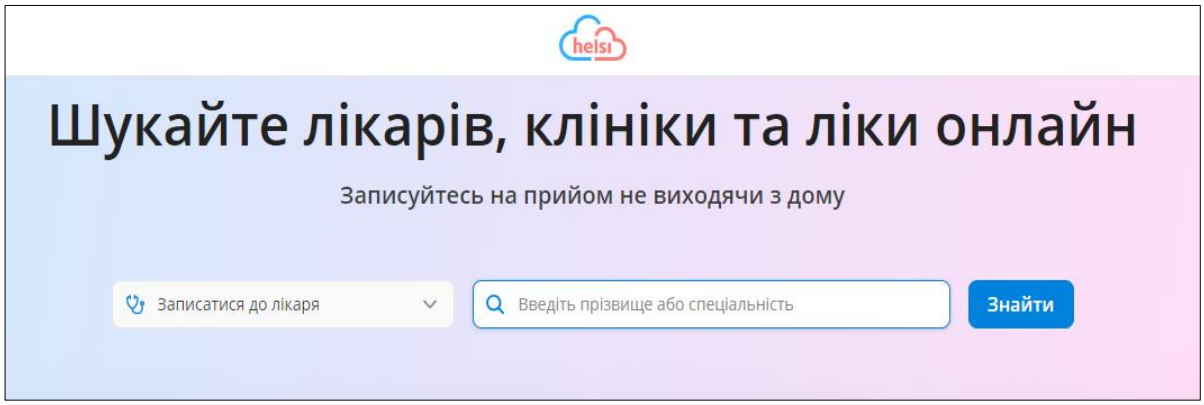

Рисунок 1.1 – Головна сторінка застосунку «Helsi» [2]

Також розглянемо ще одну систему «Поліклініка без черг» (рис. 1.2).

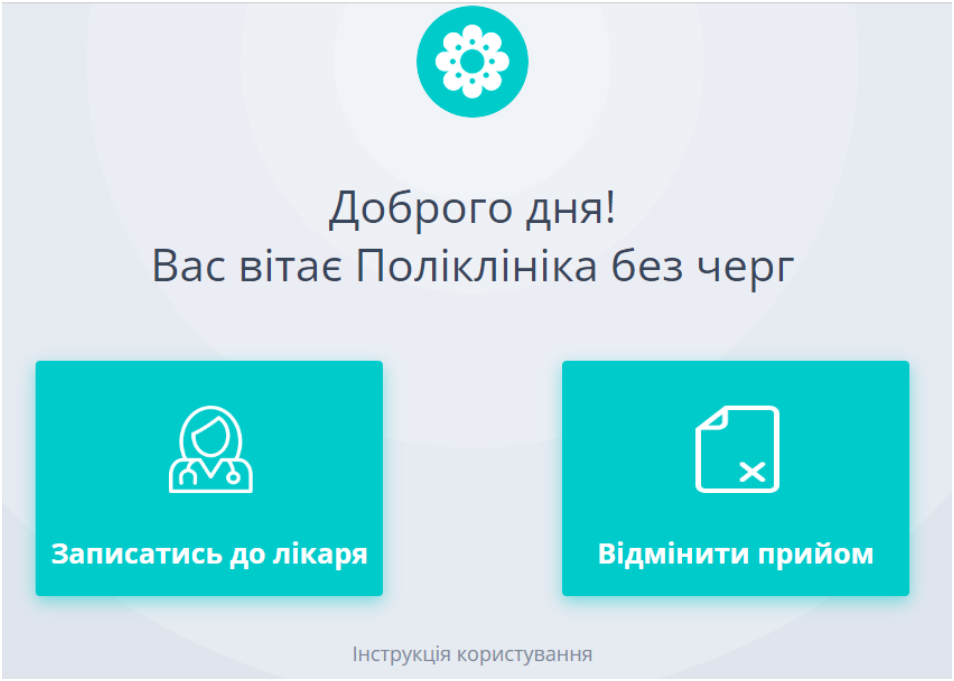

Рисунок 1.2 – Головна сторінка за стосунку «Поліклініка без черг» [3]

Дана система управління потоком користувачів-пацієнтів та автоматизації в лікувальних закладах відрізняється простотою впровадження та легкістю у користуванні.

Системи-аналоги є досить популярними, однак, теж мають свої недоліки.

Майбутній розроблений програмний продукт матиме частково схожі функції, однак, включатиме і додаткові можливості, що відсутні в аналогах.

З розвитком ІТ-технологій зростає складність інформаційних систем і обсягів даних, що в них зберігаються. Кожна прикладна програма відображає певну частину реального світу та містить його наявний опис у вигляді даних. Великі обсяги відповідних даних зберігаються окремо від коду програми, що виконується, і організовуються у вигляді баз даних. Для роботи з цими даними використовуються спеціальні програмні комплекси, відомі як системи управління базами даних (СУБД) [4].

Тепер на сучасному етапі покращення засобів обробки даних переважає клієнт-серверна архітектура. У цьому контексті прикладна програма може взаємодіяти з іншими програмами в мережі через сервер баз даних, обмінюючись з ним даними.

Технологія клієнт-сервер передбачає таку взаємодії програмних компонентів, при якій вони працюють як єдина система. Як випливає з назви, існує клієнтський процес, що запитує певні ресурси, і серверний процес, що ці ресурси надає. Вони не обов’язково повинні бути на одному комп’ютері. Зазвичай сервер розташовується на одному вузлі локальної мережі, а клієнти – на інших вузлах [5].

У контексті баз даних клієнт відповідає за користувацький інтерфейс і логіку роботи, функціонуючи як робоча станція. Він отримує запит від користувача, перевіряє його синтаксис і формує запит до бази даних мовою SQL або іншою відповідною мовою, згідно з логікою клієнтської програми. Потім клієнт надсилає запит серверу, чекає на відповідь і форматує отримані дані для відображення користувачу. Сервер приймає та обробляє запити до бази даних, а потім надсилає результати назад клієнту. Це включає перевірку

повноважень клієнта, забезпечення цілісності даних, виконання запиту та оновлення даних. Додатково сервер управляє паралельністю процесів і відновленням даних [5].

Архітектура клієнт-сервер має кілька переваг:

– існує ширший доступ до наявних баз даних;

– зростає загальна продуктивність системи, оскільки клієнти та сервери розміщені на різних комп'ютерах і можуть виконувати завдання паралельно. Це спрощує налаштування продуктивності сервера, який відповідає лише за роботу з базою даних;

– знижується вартість апаратного забезпечення. Достатньо потужного комп’ютера для сервера, що відповідає за зберігання і управління базою даних;

– зменшуються комунікаційні витрати, оскільки частину операцій виконують клієнтські комп’ютери і посилають лише запити до баз даних через мережу, що дозволяє значно зменшити обсяг пересилання даних;

– підвищується рівень несуперечливості даних, оскільки кожна клієнтська програма не потребує власної перевірки цілісності даних;

– архітектура клієнт-сервер природно відображається на архітектурі відкритих систем [6].

1.2 Постановка завдання на кваліфікаційну роботу бакалавра

Головною метою при розробці системи – спрощення та автоматизація процесу прийому пацієнтів лікарями. Така система допоможе формалізувати та оптимізувати рутинний процес запису та очікування своєї черги пацієнтами, тим самим найбільше піднявши рівень задоволеності клієнтів, які відвідали лікарню. Також важливим завданням розробки буде забезпечення безпеки користувачів шляхом збільшення фізичної дистанції між пацієнтами та мінімізації близьких контактів один з одним.

Для реалізації клієнт-серверного додатку необхідно виконати наступні завдання:

– проаналізувати предметну область;

– провести дослідження існуючих аналогів;

– навчитися працювати з Python фреймворк Django;

– навчитися працювати з Django Rest Framework;

– реалізувати систему авторизації та реєстрації;

– реалізувати моделі «Лікаря», «Пацієнта» та «Адміністратора системи»;

– для керування системою реалізувати клієнт-серверний додаток;

– реалізувати API сервісу;

– ознайомитися з роботою фреймворку UIKit для створення елементів інтерфейсу користувача мобільного додатку;

– реалізувати можливість потрапляння в чергу пацієнтів за допомогою зчитування QR коду користувачем мобільного додатку;

– реалізувати механізм отримання даних додатком;

Висновки до розділу 1

У першому розділі роботи було проведено аналіз проблематики дослідження, а також популярних систем управління чергами. Здебільшого ринок наповнений системами термінального типу, що використовує термінал реєстрації як пристрій організації черги. Такі системи мають багато мінусів, таких як відсутність контролю з боку адміністрації, неможливість ідентифікації клієнтів, неконтрольований запис.

РОЗДІЛ 2
СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення

Для розробки програмного забезпечення поставимо ряд вимог, а саме:

– функціональні вимоги:

1) реєстрація пацієнта та лікаря в системі;

2) авторизація адміністратора та лікаря в системі, використовуючи веб-інтерфейс;

3) авторизація пацієнта в системі за допомогою мобільного додатку;

4) контроль адміністратором реєстрації лікарів;

5) можливість реєстрації лікарів адміністратором;

6) можливість зміни лікарем становища пацієнтів у черзі;

7) реєстрація пацієнтів на прийом до лікаря за допомогою сканування QR-коду;

8) відстеження черги пацієнтами;

9) отримання пацієнтами очікуваного прийому;

10) отримання списку прийомів лікарем та адміністратором системи;

11) отримання даних системи API;

– вимоги до зручності і простоти використання:

1) система повинна мати інтуїтивно зрозумілий веб-інтерфейс, написаний за допомогою технології Bootstrap;

2) система повинна мати мобільний інтерфейс, написаний з використанням бібліотеки UIKit;

3) інтерфейс має бути інтуїтивно зрозумілим для середньостатистичного користувача за першого ж прецеденту;

– обмеження проектування:

1) операційні системи: MacOS X версії Big Sur 11.1; Apple iOS версії 17.0;

2) мови: Python, Swift;

3) додаткові бібліотеки:

– Python:

1) beautifulsoup4 – бібліотека для збору інформації з веб-сторінок;

2) bootstrap4 – бібліотека для доступу до набору інструментів;

3) створення красивого фронтенду;

4) dj-database-url – бібліотека версії 0.5.0 дозволяє використовувати змінні середовища в шаблонах django;

5) django-bootstrap-datepicker-plus – бібліотека версії 3.0.5 дозволяє отримати доступ до bootstrap – інструменту вибору дати та часу користувачами;

6) django-crispy-forms – бібліотека версії 1.11.2 – дозволяє обробляти власні форми з використанням bootstrap4;

7) django-extra-fields – бібліотека версії 0.3 – розширює можливості створення моделей БД системи;

8) django-jquery – бібліотека версії 3.1.0 – дозволяє отримати доступ до функціоналу jQuery;

9) django-widget-tweaks – бібліотека версії 1.4.8 – дозволяє налаштувати відображення полів форми в шаблонах;

10) djangorestframework – бібліотека версії 3.12.4, надає доступ до набору інструментів для створення API REST;

11) gunicorn – бібліотека версії 20.1.0 дозволяє створити WSGI сервер для використання у UNIX-системах;

12) html5lib – бібліотека версії 1.1, призначена для синтаксичного аналізу у HTML;

13) Pillow – бібліотека версії 8.2.0, призначена для рендерингу та обробки зображень;

14) psycopg2 – бібліотека версії 2.8.6, призначена для адаптації бази даних при розгортанні сервера у хмарі Heroku;

15) PyPDF2 – бібліотека версії 1.26.0, призначена для створення PDF-документів із HTML-сторінок;

16) python-bidi – бібліотека версії 0.4.2, призначена для роботи декодування UTF-8;

17) pytz – бібліотека версії 2021.1, призначена для роботи з часом;

18) qrcode – бібліотека версії 6.1, призначена для генерації QR-кодів;

19) reportlab – бібліотека версії 3.5.66, виконує функцію генерації PDF;

20) six – бібліотека версії 1.15.0, допомагає поєднувати Python версії 2 та 3;

21) soupsieve – бібліотека версії 2.2.1, призначена для забезпечення вибору, зіставлення та фільтрації з використанням сучасних селекторів CSS;

22) sqlparse – бібліотека версії 0.4.1, забезпечує підтримку синтаксичного аналізу, поділу та форматування операторів;

23) SQL;

24) webencodings – бібліотека версії 0.5.1, реалізує Python – стандарт кодування WHATWG;

25) whitenoise – бібліотека версії 5.2.0, призначена для зберігання та обслуговування статичних даних системи у хмарі;

26) xhtml2pdf – бібліотека версії 0.2.5, призначена для перекладу HTML у PDF;

– Swift:

1) Alamofire – мережева бібліотека для роботи мобільного додатку з HTTP;

2) SwiftyDate – бібліотека для роботи з датами;

3) SwiftyJSON – бібліотека для серіалізації та десеріалізації даних у JSON формат.

Веб-застосунок буде розроблений за моделлю клієнт-сервер, де клієнт ініціює HTTP-запит до сервера для отримання певних даних з відповідної частини сервера. UML-діаграма клієнт-серверного додатку зображена на рисунку 2.1.

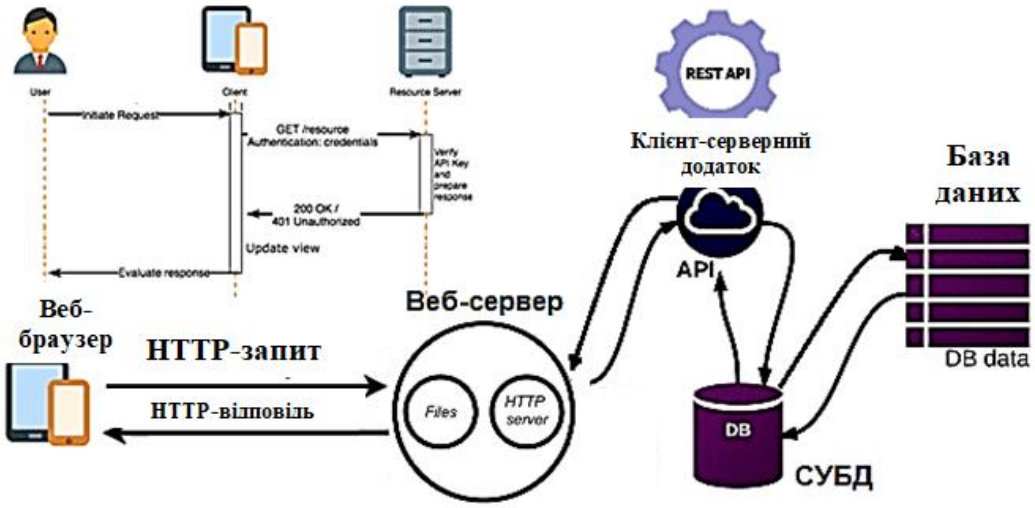

Рисунок 2.1 – UML-діаграма клієнт-серверного додатку [7]

Система, що розробляється, має клієнт-серверну архітектуру, яка визначає загальні принципи організації взаємодії в мережі та включає в себе такі особливості [8]. Користувач надсилає певний запит на сервер, де той обробляється та надсилає відповідь користувачу. Досить хорошою можливістю подібної архітектури є можливість одноразового обслуговування кількох клієнтів. Система складається із чотирьох пов’язаних частин (рис. 2.2).

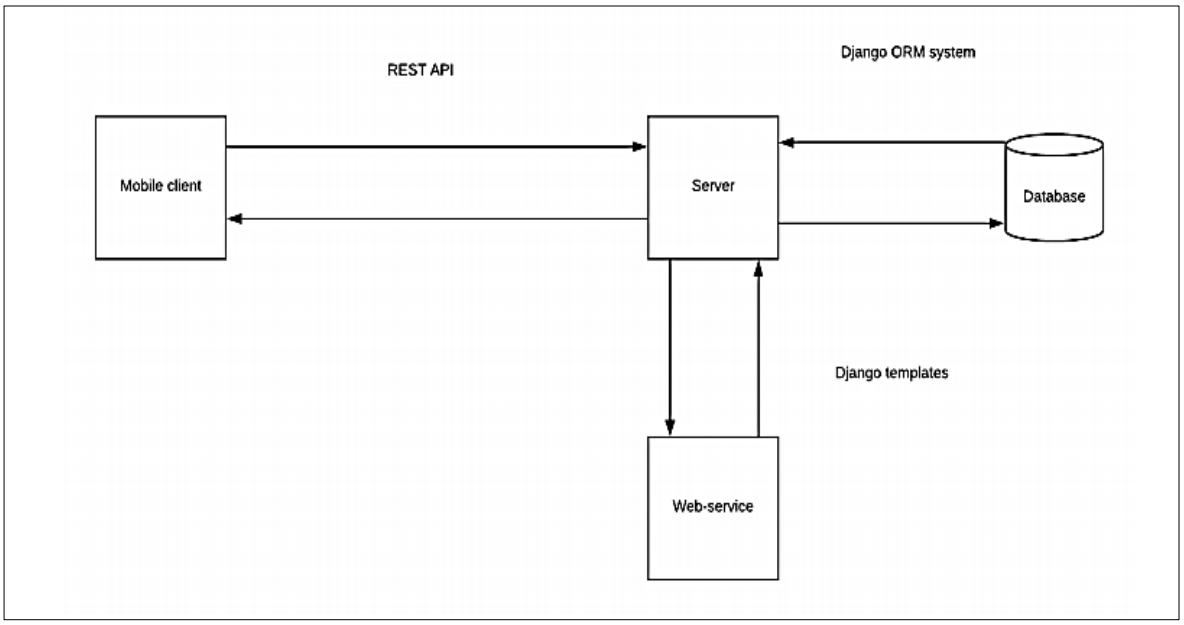

Рисунок 2.2 – Архітектурна схема системи

Перша та основна частина всього проекту – це сервер. Під сервером розуміємо машину, здатну отримає HTTP-запит і обробивши його, повернути коректну відповідь [9]. Також сервер реалізує параметри зберігання, захисту та доступу до даних.

База даних, друга важлива частина системи, являє собою організовану структуру, призначену для обробки, зберігання та зміни інформації. Для зв'язку із сервером використовується механізм Django ORM, який дозволяє ефективно працювати з моделями бази даних [10].

Ще однією важливою частиною проекту є веб-інтерфейс, що надає адміністратору та лікарям доступ до взаємодії з системою. Саме за допомогою нього, лікарі зможуть отримувати дані про стан та склад черг, а адміністратори керувати моделями бази даних. Для зв'язку веб-інтерфейсу із сервером використовується динамічна генерація шаблонів фреймворку Django [11].

Четвертою частиною проекту є мобільний додаток пацієнта. Для взаємодії між сервером і клієнтом використовується архітектурний підхід REST API. За допомогою формування JSON запитів, сервер обмінюється інформацією із клієнтом. Обмін інформації відбувається завдяки потужному та гнучкому фреймворку Django Rest Framework [12].

2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання

При розробці продукту використовувані технології та інструменти реалізації можна поділити на дві групи. Технології, що використовуються для розробки мобільної частини системи та технології використовуються для створення веб-додатку.

Для створення мобільної частини планується використовувати мову програмування Swift. Це мова, створена компанією Apple спеціально для розробки під мобільну ОС iOS. Для розробки елементів інтерфейсу користувача передбачається використання популярного фреймворку UIKit [13]. А для

роботи мобільного додатку з мережевим стеком вибрано HTTP мережеву бібліотеку Alamofire, яка дозволить виконувати основні мережеві завдання, такі як завантаження файлів або запит даних із сторонніх RESTful API. Для серіалізації та десеріалізації об’єктів планується використання фреймворку SwiftyJSON, який створений для усунення optional chaining.

Веб-програма реалізується мовою програмування Python версії 3.6 з допомогою фреймворку Django. Вибір Django для розробки веб-програми не випадковий, завдяки таким характеристикам як: поділ візуальної частини та бізнес-логіки, здатності до легкої та швидкої розширюваності, розвиненої інфраструктури, вираженої у великій кількості бібліотек та плагінів, Django стає ідеальним інструментом для створення веб-додатків клієнт-серверного типу. Для створення гнучкого та потужного Web API сервісу вибрано бібліотеку Django Rest Framework.

Головним завданням DRF є перетворення класів Model у формат Json для подальшої передачі через API.

Для створення інтерфейсу користувача веб-програми використовується Bootstrap набір інструментів, який включає готові HTML та CSS шаблони.

При розробці мобільного iOS додатку використовуватиметься шаблон проектування Model-View-Controller, який дає об’єктам одну з трьох ролей: model, view або controller. Відповідно до документації Apple кожен із трьох типів об’єктів відділений і пов’язаний з іншими абстрактними законами. Завдяки MVC підходу, створюється чітка ієрархія взаємодії об’єктів один з одним, що дає можливість використовувати багато разів створені у додатку об’єкти. Також MVC програми простіше і зручніше масштабувати.

При розробці веб-частини сервісу реалізується архітектурний патерн Django – ModelView-Template, який за фактом є модифікацією патерну MVC, за допомогою якого розробляється мобільний iOS додаток. Основними елементами патерну є:

– URL dispatcher – вибір ресурсів для обробки запитів на основі запитаної URL-адреси;

– Model – опис даних, які використовуються в програмі;

– View – отримання, обробка та відповідь на запити користувача;

– Template – логіка подання у вигляді згенерованої HTML розмітки. У MVC цьому компоненту відповідає View.

Web API сервісу, розроблений засобами DRF, складається з трьох шарів. Перший шар – це серіалізатор, завдання якого перетворювати інформацію, що зберігається в базі даних, в JSON формат, який успішно передається через API. Другим шаром є ViewSet, що визначає CRUD API функції. Третім шаром є маршутиризатор, завдання якого визначати URL-адресу з доступом до ViewSet.

Програма використовує вільну реляційну систему управління базами даних – MySQL (рис. 2.3).

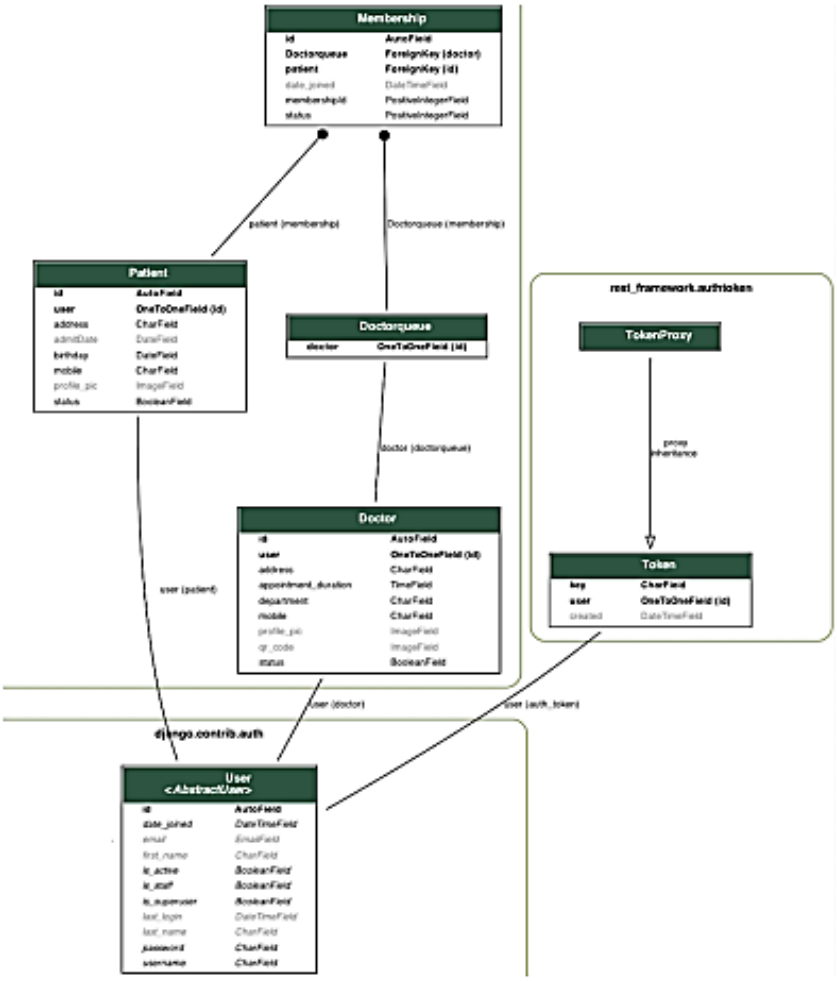

Рисунок 2.3 – Схема моделі бази даних [14]

Нижче опишемо компоненти БД:

– User – являє собою модель абстрактного користувача з полями: id, date_joined, password, username, is_superuser. Поле id – порядковий номер користувача в системі, date_joined – час реєстрації користувача, password – пароль, username – ім’я користувача, is_superuser – перевірка на адміністратора всієї системи;

– Token – jwt токен автентифікації користувача;

– Doctor – модель лікаря, успадковується від User і має поля: id, address, appointment_duration, department, mobile, profile_pic, qr_code та status. Поле id – порядковий номер лікаря у системі, address – адреса лікаря, department – департамент лікаря, mobile – номер телефону, profile_pic – зображення профілю, qr_code – код, що генерується, асоційований з id лікаря, appointment_duration – середній час прийому лікарем одного пацієнта;

– Patient – модель пацієнта, успадковується від User та має поля: id, address, birthday, mobile, profile_pic, admitDate. Поле id – порядковий номер пацієнта в системі, addres – адреса пацієнта, mobile – номер телефону, profile_pic – зображення профілю, admitDate – дата реєстрації пацієнта;

– Doctorqueue – модель, що представляє чергу в системі, пов’язана з моделлю Doctor – зв’язком один до одного і моделлю Membership зв’язком багато-багатьом;

– Membership – модель членства у черзі. Має поля: id – порядковий номер в системі, date_joined – час потрапляння в чергу, membershipId – номер потрапив у чергу, Doctorqueue для зв’язку з Doctorqueue та patient для зв’язку з моделлю patient. Поле status показує поточний статус пацієнта у черзі (відвідав/очікує прийому/вийшов).

Висновки до розділу 2

У другому розділі було сформульовано вимоги до системи, що включають функціональні вимоги, вимоги до зручності і простоти

використання, обмеження проектування та вимоги стосовно Python та Swift. Крім того, була описана архітектура програми, подана технічна інформація про інструменти розробки, були коротко описані використовувані мови програмування та бібліотеки, а також розглянуті їх переваги та недоліки. Також було дано детальну інформацію про моделі бази даних системи.

РОЗДІЛ 3
РОЗРОБКА КЛІЄНТ-СЕРВЕРНОГО ДОДАТКУ ДЛЯ ОРГАНІЗАЦІЇ ЕЛЕКТРОННОЇ ЧЕРГИ

3.1 Практична реалізація об’єкта проектування

Обґрунтувавши вимоги до розроблюваного програмного забезпечення та проектування програмного забезпечення, а також детально описавши засоби, методи і алгоритми вирішення поставленого завдання, – опишемо етапи функціонування розробленого клієнт-серверного додатку.

Можливість реєстрації у веб-додатку, передбачена лише для моделі «лікарі». Дані «адміністратора», який також працює у веб-додатку, задаються при розгортанні системи в хмарі. Новому лікарю необхідно заповнити форму реєстрації, що містить: Ім’я, Прізвище, Логін, Пароль, Мобільний телефон, Фото профілю, Середній час прийому пацієнта (рис. 3.1).

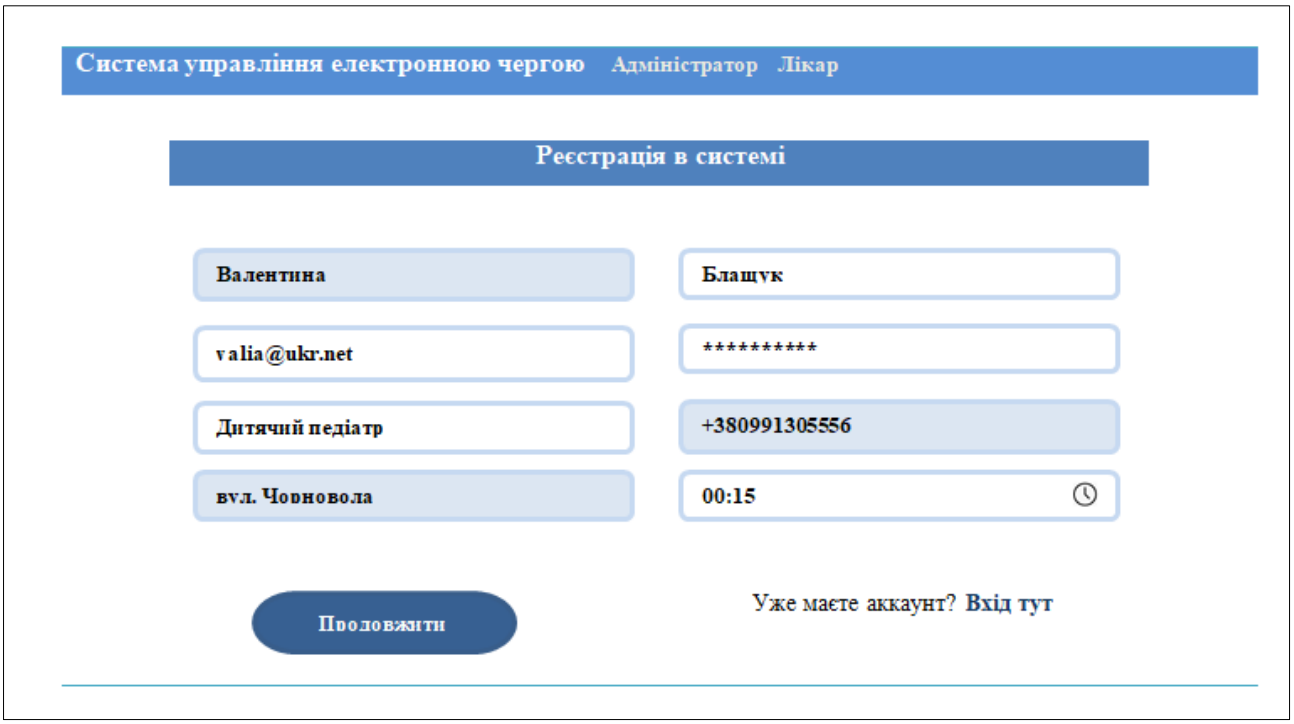

Рисунок 3.1 – Сторінка реєстрації лікаря

Натиснувши кнопку «Продовжити», користувач потрапляє на сторінку перевірки (рис. 3.2).

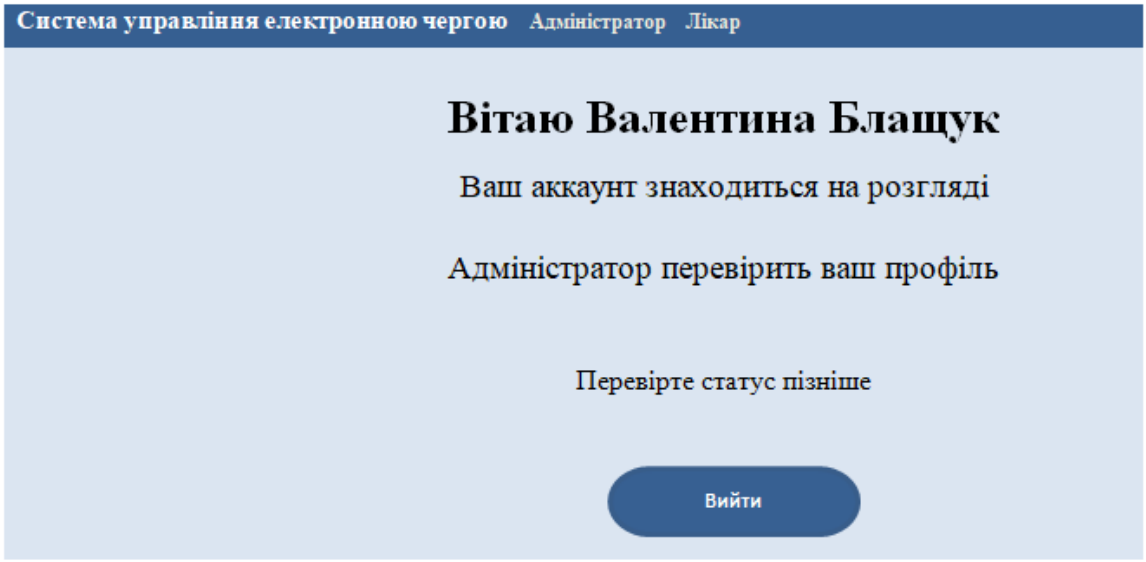

Рисунок 3.2 – Сторінка перевірки статусу

Доступ до основного функціоналу програми, новий лікар отримає тільки після схвалення адміністратором системи, тоді ж система присвоює йому QR-код, за допомогою якого пацієнти зможуть потрапити до нього на прийом.

Авторизація в системі відбувається за допомогою введення в bootstrap форму логіну і пароля. Введені дані перевіряються на коректність і якщо помилки відсутні, користувачів перенаправляють до особистого кабінету.

Далі опишемо можливості адміністратора системи.

Авторизувавшись, адміністратор отримує доступ до управління всією системою.

Особистий кабінет містить 4 розділи: «Статистика», «Лікарі», «Пацієнти», «Прийоми». У будь-який момент часу лікар може вийти з системи, натиснувши на кнопку «Вийти».

На сторінці зі статистикою адміністратор може побачити: кількість лікарів, які здійснюють прийом; кількість лікарів, які чекають на підтвердження; кількість зареєстрованих пацієнтів; кількість успішних

прийомів і, які не відбулися; кількість людей, які чекають своєї черги в даний час (рис. 3.3).

Рисунок 3.3 – Сторінка адміністратора

На сторінці «Лікаря» адміністратор може переглянути список лікарів, зареєструвати та схвалити реєстрацію нового лікаря (рис. 3.4).

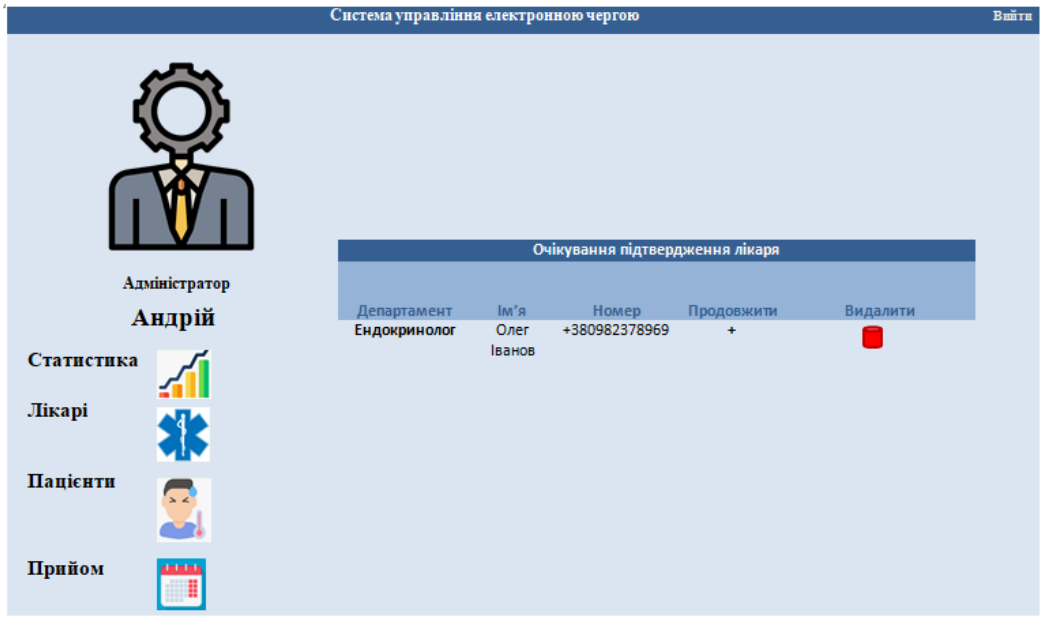

Рисунок 3.4 – Сторінка підтвердження інформації про лікаря

Натискання адміністратором на кнопку «Підтвердити» ініціює генерацію індивідуального QR-коду, асоційованого із підтвердженим лікарем. Генерації QR-кодів здійснюється за допомогою бібліотеки Python qrcode на вхід якої надходить id схваленого лікаря. Метод qrcode.make() створює об’єкт PilImage, який передається в функцію paste класу Image. Після створення QR-коду система заносить запис про новий лікарі у БД.

Адміністратор, перейшовши до пункту меню «Список лікарів», може побачити список усіх зареєстрованих у системі лікарів, а також усю доступну інформацію, асоційовану з конкретним лікарем, включаючи QR-коди (рис. 3.5).

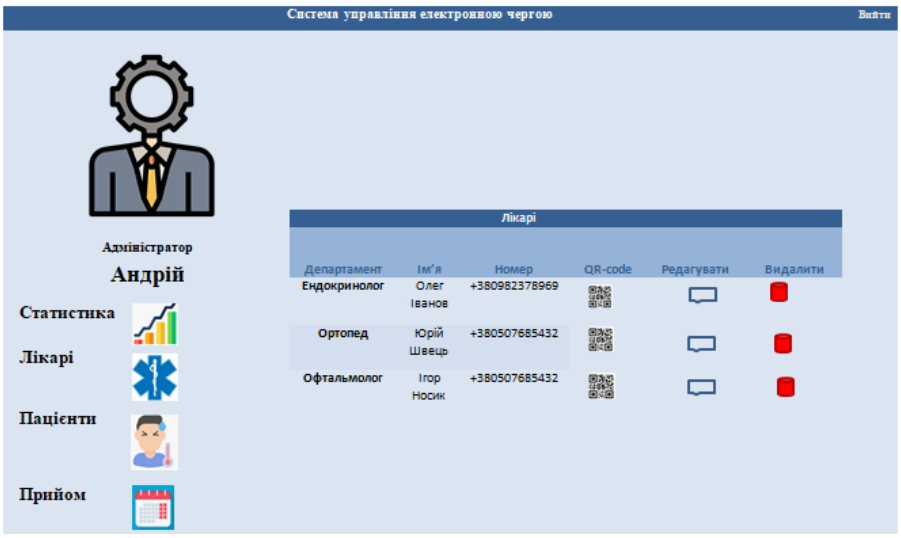

Рисунок 3.5 – Сторінка з інформацією про лікарів

На сторінці «Пацієнти» адміністратор може переглянути список наявних пацієнтів та за необхідності зареєструвати нового.

Сторінка «Прийоми» містить інформацію про всі прийоми пацієнтів лікарями. Також адміністратор може виписати рахунок, на основі успішного відвідування пацієнтом лікаря. Для створення рахунку у форматі PDF використовується фреймворк xhtml2pdf. Макет рахунку створюється в HTML з кодуванням UTF-8, далі він передається в метод render_to_pdf, який за допомогою pisa декодується і записується в новий файл PDF формату. Для того,

щоб pisa зміг використати кирилицю необхідно обов’язково вказати у шаблоні шлях до кастомного шрифту.

3.2 Тестування та налагодження інформаційно-комп’ютерної системи

Розглянемо роботу програмного продукту та опишемо тестування його продуктивності.

Схвалений адміністратором лікар отримує доступ до особистого кабінету, на головній сторінці якого він може побачити наданий йому QR-код, а також статистику прийомів, що включає: кількість очікуваних, прийнятих та залишених чергу пацієнтів (рис. 3.6).

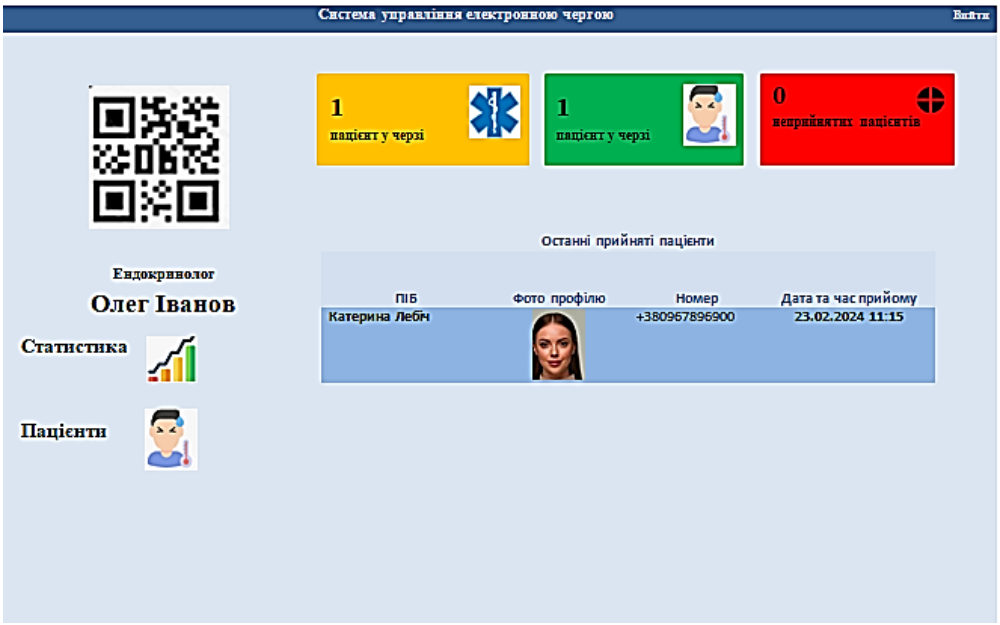

Рисунок 3.6 – Особистий кабінет лікаря

Лікар, перейшовши до пункту меню «Пацієнти» може побачити списки прийнятих та неприйнятих пацієнтів, і навіть перейти на сторінку управління чергою.

У меню «Управління чергою» лікар може потрапити, перейшовши до пункту «Прийом пацієнтів». Перед лікарем відкриється таблиця з усіма

пацієнтами, які перебувають у черзі. Пацієнт, прийом якого здійснюється в даний момент часу, виділений у спеціальне поле.

Усі записи пацієнтів зберігаються у моделі Membership БД системи. Membership має поле status, яке може зберігати одне зі значень прапорів: 0, якщо пацієнт перебуває зараз у черзі; 1, якщо пацієнт знаходиться в даний момент часу на прийомі; 2, якщо пацієнт залишив чергу або його виключив лікар, що здійснює прийом. Модель черги до лікаря (Doctorqueue) утворюється з набору Membership, тобто з усіх записів асоційованих із цією чергою. Порядок усередині черги утворюється завдяки полю membershipId моделі Membership, саме в ньому зберігається номер пацієнта в черзі.

Лікар, натисканням на кнопку «Завершити прийом» переводить запис пацієнта в розділ «Прийняті пацієнти» (змінює значення поля status моделі Membership с 1 на 2) та ініціює виклик наступного пацієнта зі списку (змінює значення поля status моделі Membership наступного користувача з 0 на 1).

Лікар здатний переміщати пацієнтів усередині черги натисканням на кнопки «Підняти» або «Опустити», змінюючи тим самим поле membershipId моделі Membership, пов’язаної з конкретним користувачем (рис. 3.7).

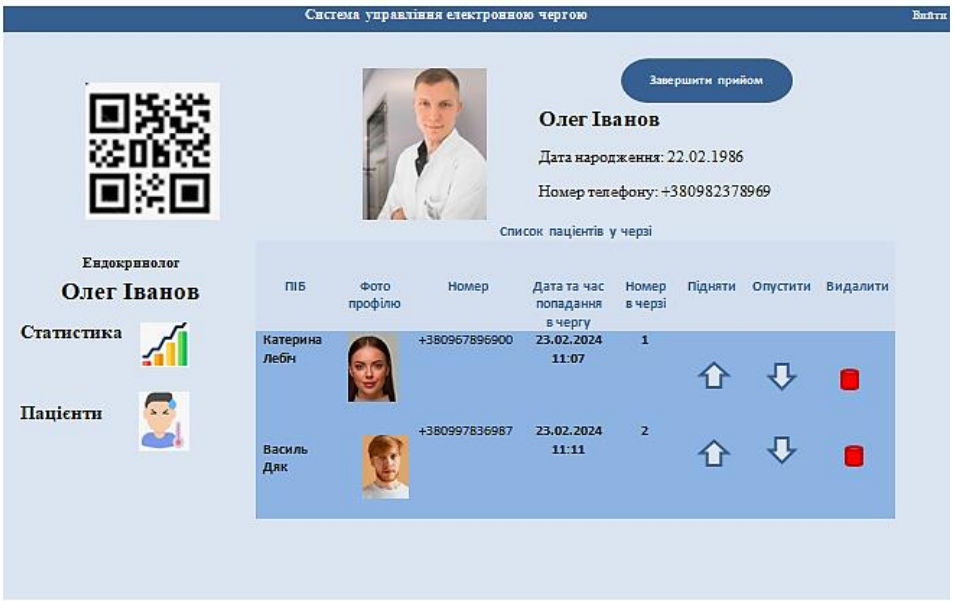

Рисунок 3.7 – Сторінка управління чергою

Перейдемо до стадії тестування. Тестування продуктивності охоплює сукупність ключових дій, що здійснюються на різних стадіях розробки. Кожна дія має свої особливості та завдання, які необхідно виконати.

Браузер (клієнт) ініціює діалог із сервером через протокол HTTP, надсилаючи запит із зазначенням, яку інформацію та в якому форматі він очікує отримати. Після отримання та обробки запиту сервер відповідає браузеру повідомленням (HTTP-відповіддю), що містить запитані дані (зазвичай у вигляді HTML-документа). Ці повідомлення включають корисну інформацію та спеціальні коди стану, за якими браузер може визначити, чи був запит успішно оброблений.

Результат обробки запиту, який вказаний у першому рядку відповіді сервера, називається кодом відповіді або кодом стану. Ці коди відправляються щоразу під час взаємодії браузера з сервером, навіть якщо користувач сайту їх не бачить.

Щоб дізнатися код стану запиту в браузері, відкриємо інструменти веб-розробника. Значення коду можна знайти в стовпці «Status» на вкладці «Network». Після відкриття консолі необхідно оновити сторінку (рис. 3.8).

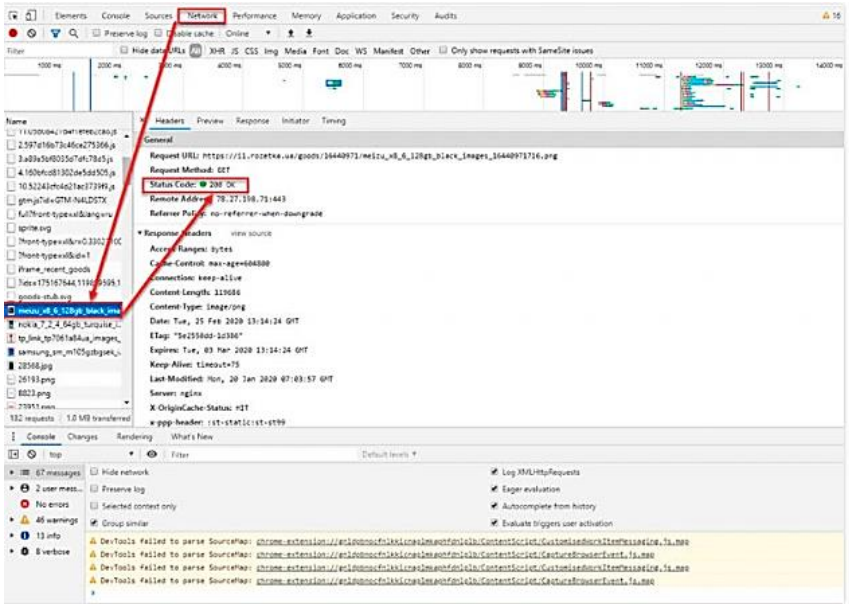

Рисунок 3.8 – Результат обробки запиту коду стану

При тестуванні програмного інтерфейсу додатку (API) використовуються запити HTTP. У відповідь на ці запити сервер надсилає різні коди, які можуть вказувати на помилку або інформувати про стан сервера. API надає користувачам ці дані. Процедура обміну інформацією та формат передачі даних структуровані таким чином, щоб обидві сторони знали, як взаємодіяти між собою.

За допомогою API можна виявити такі помилки в програмному забезпеченні:

– проблеми з безпекою API;
– невірно структуровані відповіді (JSON або XML);
– некоректна обробка даних;
– відсутність або дублювання функціоналу;
– питання багатопоточності;
– проблеми з продуктивністю.

Коди, що пов’язані з функціональною роботою ресурсу, вказують на стан сторінок. Інформація, яку кожен код передає після запиту користувача, дозволяє серверу змінювати обробку документа. У випадку помилки знання результату запиту дозволяє виправити її. Діагностика кодів помилок також допомагає виявити слабкі місця програмного забезпечення.

3.3 Захист інформаційно-комп’ютерної системи

Розглянемо, для прикладу, особливості організації механізмів авторизації, автентифікації у мобільному додатку. Реєстрація та авторизація в мобільному додатку здійснюється за допомогою AuthViewController та RegisterViewController. При реєстрації пацієнту необхідно вказати ім’я, прізвище, дату народження та номер телефону, вибрати логін та призначити пароль. Для безпеки було вирішено використовувати систему «токенів доступу» – ключів, визначених системою в єдиному екземплярі, за якими

сервер може однозначно визначити користувачів при кожному надісланому запиті (рис. 3.9).

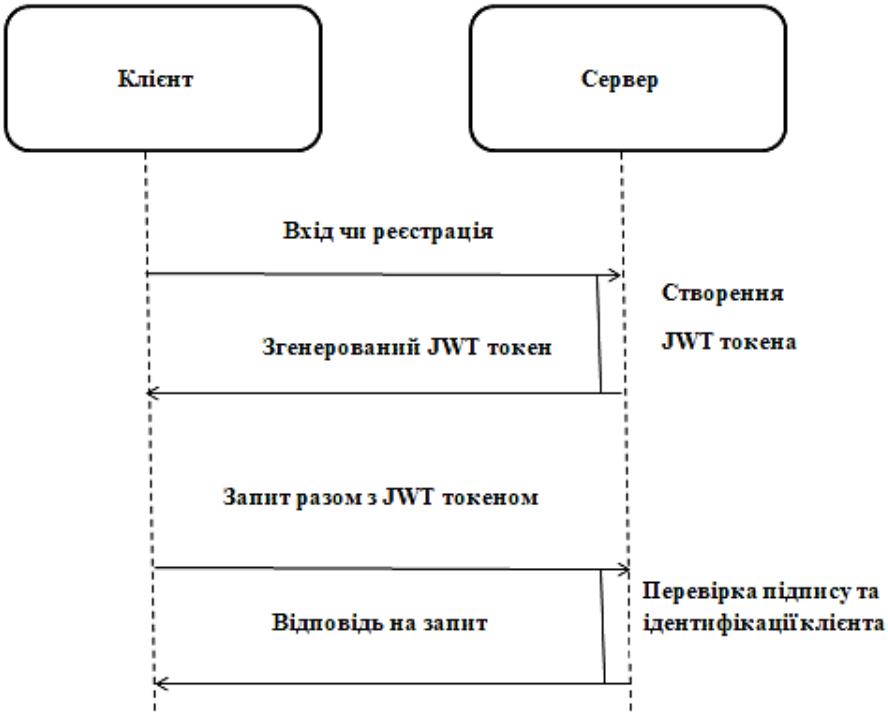

Рисунок 3.9 – Ілюстрація роботи JWT токена

Завдяки системі JWT токенів сервер може регулювати час користувача сесії, без повторних авторизацій. Повторний вхід до системи – оновлює існуючий токен користувача. Для створення JWT токенів на сервері використовували вбудовані функції бібліотеки rest_framework.

Користувачеві мобільного додатку в особистому кабінеті доступні такі функції: сканування QR-коду, перегляд особистих даних, перегляд списку відвіданих черг.

Для сканування QR-коду в мобільному додатку використовується функціонал swift бібліотеки AVFoundation, код якого знаходиться у файлі QRReaderViewController.swift. Отримавши відскановане значення номера черги, програма робить POST запит на додавання пацієнта до черги. Якщо значення номера черги не збігається з існуючими на сервері, пацієнту приходить Alert повідомлення про невалідність даних. Також система блокує можливість

потрапляння пацієнта в чергу, в якій він уже знаходиться, пацієнт у такому разі отримує повідомлення про помилки.

У разі успішної реєстрації в черзі, на екрані зі списком черг з’являється новий запис. Дані записи передають пацієнтові інформацію про поточний стан черги: порядковий номер, очікувана кількість часу, що залишився почекати, час з моменту реєстрації в черзі. Пацієнт у будь-який момент часу, може залишити чергу натиснувши кнопку «Вийти з черги» ініціюючи цією дією зміну поля status у моделі Membership.

Висновки до розділу 3

У даному розділі описано етапи функціонування розробленого клієнт-серверного додатку, обґрунтовано особливості реалізації клієнт-серверного додатку через мережу Інтернет та мобільної програми, представлені елементи інтерфейсу користувача з описом їх функціональності, показано та описано етапи тестування системи, а також означено особливості організації механізмів авторизації та автентифікації.

ВИСНОВКИ

Протягом квітня 2024 року у США, Великій Британії та Канаді швидко поширюється новий штам коронавірусу FLiRT, який є похідним від субваріанту «Омікрон – JN.1», також відомого як «Дженні». Найбільш відомим серед штамів FLiRT є варіант KP.2. Про це повідомили фахівці Школи громадської охорони здоров'я Університету Джонса Гопкінса у США.

Тому для убезпечення людей на майбутнє постало питання розробки додатку з управління чергами у закладах охорони здоров'я. Варто зазначити, що розробка такої система надасть безліч плюсів, як організації, так її клієнтам. Для відвідувачів процедура очікування своєї черги стане зрозумілою та прогнозованою, що, у свою чергу, знизить рівень їхньої нервозності та убезпечить від контакту з хворою людиною. Для організації – відкриває нові можливості при ухваленні рішень.

Тому для реалізації клієнт-серверного додатку було вирішено наступні завдання:

– проаналізовано предметну область, де означено стан проблематики зараження коронавірусною інфекцією та означено шляхи мінімізації ризиків контактів між пацієнтами завдяки дотриманню дистанції та попереднього запису на прийом;

– проведено дослідження існуючих аналогів, таких як «Helsi» та «Поліклініка без черг», де означено їх найважливіші функції у порівнянні з розроблюваною системою;

– проведено роботу з використанням у Python фреймворку Django, завдяки чому бібліотека версії 3.0.5 дозволяє отримати доступ до інструменту вибору дати та часу користувачами; бібліотека версії 1.11.2 – дозволяє обробляти власні форми з використанням bootstrap4; бібліотека версії 0.3 – розширює можливості створення моделей БД системи; бібліотека версії 3.1.0 – дозволяє отримати доступ до функціоналу jQuery; бібліотека версії 1.4.8 – дозволяє налаштувати відображення полів форми в шаблонах;

– проведено роботу з використанням Django Rest Framework, завдяки якому бібліотека версії 3.12.4 надає доступ до набору інструментів для створення API REST;

– реалізовано систему авторизації та реєстрації, що здійснюється за допомогою AuthViewController та RegisterViewController;

– реалізовано моделі «Лікаря», «Пацієнта» та «Адміністратора системи», кожен з яких має власні розширені функціональні можливості, що дозволяють оптимізувати процес роботи у клієнт-серверному додатку;

– для керування системою було реалізовано клієнт-серверний додаток, що вирішує проблему організації соціального дистанціювання у місцях підвищеної небезпеки поширення небезпечних захворювань, таких як COVID-19, за рахунок зменшення ймовірностей можливих фізичних контактів, заражених з неінфікованими людьми. Також додаток допомагає формалізувати та оптимізувати управління потоками відвідувачів, допомогає підвищити якість обслуговування відвідувачів лікарень шляхом автоматичного розподілу потоків відвідувачів та організації контролю та обліку співробітників, які здійснюють обслуговування;

– реалізовано API сервісу, завдяки якому існує взаємодія між сервером і клієнтом з використанням архітектурного підходу;

– здійснено роботу з фреймворком UIKit, що використовувався для розробки елементів інтерфейсу користувача;

– реалізовано можливість потрапляння в чергу пацієнтів за допомогою зчитування QR коду користувачем мобільного додатку, де генерації QR-кодів здійснюється за допомогою бібліотеки Python qrcode на вхід якої надходить id схваленого лікаря. Після створення QR-коду система заносить запис про нового лікаря у БД;

– реалізовано механізм отримання даних додатком завдяки системі API.

Розроблюваний програмний продукт пройшов тестування, під час якого відбулося налагодження системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Очікується сплеск захворюваності на FLiRT: у світі стрімко поширюється новий варіант коронавірусу. URL: https://tsn.ua/zdorovya/ochikuyut-splesk-zahvoryuvanosti-na-flirt-u-sviti-strimko-poshiryuyetsya-noviy-variant-koronavirusu-2580389.html (дата звернення: 16.05.2024).

2. Головна сторінка системи «Helsi». URL: https://helsi.me/ (дата звернення: 11.03.2024).

3. Головна сторінка системи «Поліклініка без черг». URL: https://global.newmedicine.com.ua/ (дата звернення: 11.03.2024).

4. Система керування базами даних. URL: http://surl.li/uojvn (дата звернення: 14.03.2024).

5. Сучасні клієнт-серверні технології та їх застосування при вивченні систем управління базами даних. URL: https://fi.npu.edu.ua/files/Zbirnik_KOSN/12/29.pdf (дата звернення: 12.04.2024).

6. Розуміння клієнт-серверної архітектури на прикладах. URL: https://dou.ua/forums/topic/44636/ (дата звернення: 17.05.2024).

7. Розробка та реалізація клієнт-серверного застосунку для синхронного редагування коду/документів. URL: https://ekmair.ukma.edu.ua/server/api/core/bitstreams/9d4c6244-0b59-4534-975d-8a5402b9d0e3/content (дата звернення: 18.05.2024).

8. Клієнт-серверна архітектура: матеріал з Вікіпедії. URL: http://surl.li/uojvn (дата звернення: 20.05.2024).

9. Сервер: матеріал з Вікіпедії. URL: https://uk.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%B5%D1%80 (дата звернення: 21.05.2024 р.).

10. Django ORM і QuerySets. URL: https://tutorial.djangogirls.org/uk/django_orm/ (дата звернення: 22.05.2024).

11. Django: матеріал з Вікіпедії. URL: https://uk.wikipedia.org/wiki/Django (дата звернення: 23.05.2024).

12. Django Rest Framework. URL: https://www.django-rest-framework.org/ (дата звернення: 24.05.2024).

13. Імплементуємо SwiftUI до UIKit і навпаки. URL: https://dou.ua/forums/topic/48905/ (дата звернення: 27.05.2024).

14. MySQL: матеріал з Вікіпедії. URL: https://uk.wikipedia.org/wiki/MySQL (дата звернення: 25.05.2024).