Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

БАКАЛАВРСЬКА ДИПЛОМНА РОБОТА

на тему:

Кросплатформенна інформаційно-пошукова система лікарів міста Вінниці

ПОЯСНЮВАЛЬНА ЗАПИСКА

Виконала: студенка 4 курсу, групи 1КІ-18б
напряму підготовки
123 – «Комп'ютерна інженерія»
Колеснікова Н.С.
Керівник Крупельницький Л.В.
Рецензент Лукічов В.В.

Допущено до захисту
д.т.н., проф. Азаров О.Д.
" 22 " червня 2022 р.

ВНТУ 2022

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ ДО БДР

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра обчислювальної техніки

Освітньо-кваліфікаційний рівень — бакалавр

Спеціальність — 123 «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ

Завідувач кафедри ОТ

д.т.н., професор Азаров О.Д.

"10" лютого 2022 року

З А В Д А Н Н Я

на бакалаврську кваліфікаційну роботу

Колеснікової Наталії Сергіївни

1 Тема роботи «Кросплатформенна інформаційно-пошукова система лікарів міста Вінниці» керівник роботи Крупельницький Л.В. к.т.н., доц. каф. ОТ, затверджений наказом ВНТУ від «24 березня» 2022 року № 66

2 Строк подання студентом роботи: 17.05.22

3 Вихідні дані до роботи: інформація про лікарів з відкритих джерел; графічний логотип програми; задачі інформаційно-пошукової системи; вимоги до функціональних можливостей.

4 Зміст розрахунково-пояснювальної записки: вступ, аналіз існуючих рішень та актуальність теми, проектування інформаційно-пошукового додатку; розробка кросплатформеного мобільного додатку, тестування та використання програмного додатку.

5 Перелік графічного матеріалу: презентація, діаграма послідовності пошуку, діаграма послідовності заявок.

6 Консультанти розділів роботи приведені в таблиці 1.

Таблиця 1— Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1–3	к. т. н., доцент каф. ОТ Крупельницький Л.В.		

7 Дата видачі завдання: 05 вересня 2021 року

8 Календарний план виконання роботи приведений в таблиці 2.

Таблиця 2 — Календарний план

Ч.ч	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз існуючих рішень та актуальність теми	10.09.2022	Виконано
2	Аналіз типів розробки мобільних додатків	12.09.2021—08.10.2021	Виконано
3	Аналіз існуючих рішень для розробки кросплатформенних додатків	09.10.2021—06.11.2021	Виконано
4	Аналіз системи керування бази даних.	07.11.2021—25.11.2021	Виконано
5	Проектування інформаційно-пошукового додатку	26.11.2021—22.12.2021	Виконано
6	Проектування моделі додатку	22.12.2021—04.02.2022	Виконано
7	Проектування моделі бази даних	04.02.2022—03.03.2022	Виконано
8	Розробка кросплатформеного додатку	04.03.2022—27.03.2022	Виконано
9	Тестування	27.03.2022—28.03.2022	Виконано
10	Перевірка якості виконання бакалаврської роботи та усунення недоліків	29.03.2022—9.04.2022	Виконано

Студент Колеснікова Н.С.

Керівник Крупельницький Л.В.

АНОТАЦІЯ

КолесніковаН.С. Кросплатформенна інформаційно-пошукова система лікарів міста Вінниці. Бакалаврська кваліфікаційна роботазі спеціальності 123 — Комп’ютерна Інженерія, Вінниця: ВНТУ, 2022. Пояснювальна записка містить 61 сторінки, 23 рисунків та 11 посилань.

Ця дипломна робота присвячена створенню кросплатформної інформаційно-пошукової системи лікарів у місті Вінниця. Програмна система — це додаток, який містить статичні сторінки, які є інформативними для користувача. Завдяки їм користувач отримує інформацію про наявних лікарів, актуальні медичні статті та необхідну інформацію про програму.

У цій роботі проаналізовано принципи створення та види додатків. Розроблено: модель використання системи, клієнтська та серверна частина та дизайн додатки.

Ключові слова: мобільний додаток, крос-платформна розробка, React Native.

ANNOTATION

KolesnikovaN.S. Cross-platform information retrieval system of doctors of Vinnytsia. Bachelor's thesis in the specialty 123 — Computer Engineering, Vinnytsia: VNTU, 2022. The explanatory note contains 61 pages, 23 figures and 11 references.

This thesis is devoted to the creation of a cross-platform information retrieval system of doctors in the city of Vinnytsia. A software system is an application that contains static pages that are informative to the user. Thanks to them, the user receives information about available doctors, current medical articles and the necessary information about the program.

This paper analyzes the principles of creation and types of applications. Developed: system usage model, client and server part and application design.

Keywords: mobile application, cross-platform development, React Native.

ЗМІСТ

ВСТУП

Поширеність цифрових технологій зросла більш ніж в десятки разів. Інтернет-покриття тепер доступне у всіх великих містах земної кулі. Чисельність користувачів Інтернету у 2019 році становитиме половину населення світу. А в розвинених країнах — 79% [1]. Зі зростанням технологій вся практика, у тому числі й комерційна, набула ширшого розмаху. За останнє десятиліття більшість видів операцій були інтегровані в цифрові технології.

Сьогоденна особистість докладає зусилля, щоб вирішити всі можливі справи свого життя через мережу Інтернет.За статистикою 2019 року, близько 76% населення України у віці від 16 до 55 років користуються смартфонами. А 91% учасників дослідження на 5 смартфонах користуються мобільними доповненнями. За прогнозами аналітиків, до 2023 року ця цифра перевищить 85 відсотків[2].

Таким чином, розробка мобільного додатка в сфері медицини є **актуальним** рішенням на сьогоднішній день з перспективою розвитку.

Метою дипломного проекту є розробка програмного додатка для пошуку. Згідно до мети дипломного проекту було поставлено наступні **задачі:**

— визначення актуальності предметної області та проблеми, що вирішує програмний додаток;

— огляд та аналіз сучасних технологій для розробки кросплатформенних додатків;

— аналіз типів розробки мобільних додатків;

— аналіз системи керуваннябазами даних;

— проектування інформаційно-пошукового додатку;

— проектування моделі використання додатку та моделі бази даних;

— розробка кросплатформеного додатку на фреймворку Expo;

— програмно-апаратна реалізація;

— розробка клієнтської та серверної частини;

— тестування та використання програмного додатку, використання головної сторінки додатка.

Об’єктом дослідження є процес розробки на кросплатформенних фреймворках та формування системи пошуку.

Предметом дослідження є методи та алгоритми розробки застосунків з використанням останніх фреймворків та бібліотек.

Публікація за темою роботи —"Засоби розробки кросплатформенної інформаційно-пошукової системи лікарів Вінниці"

Колесникова, Н., & Крупельницький, Л. (2022). . в НТКП ВНТУ. Факультет інформаційних технологій та комп'ютерної інженерії. Отримано з https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2022/paper/view/14881/12597

1 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА АКТУАЛЬНІСТЬ ТЕМИ

1.1 Актуальність інформаційно-пошукової системи лікарів

Сьогодні сфера медицини активно розвивається в глобальній мережі. У зв'язку з високою потребою людей на моніторинг свого здоров'я, виникають програми для пошуку та відстеження медичних послуг. При повному інформаційному навантаженні сучасній людині потрібна система для пошуку медичної допомоги. Також потребою кросплатформенного додатку є наявність різних платформ, як веб джерела, так і мобільного додатка. Тому розробка програмних рішень у цій сфері нині є актуальною.

Проект, який розробляється в рамках бакалаврської дипломної роботи, створюється для допомоги в пошуку надійного медичного спеціаліста в потрібній спеціальності лікарів сучасними людьми. Програмне рішення буде представлено у вигляді кросплатформенного додатка для звичайних користувачів. Програмний продукт надає можливості для ефективного та швидкого пошуку медичного фахівця в місті Вінниця. Користувач повинен лише мати смартфон та доступ в мережу Інтернет.

1.2 Аналіз типів розробки мобільних додатків

Виділяють такі типи мобільних додатків:

— нативні;

— web;

— кросплатформні або гібридні.

Рідні додатки це програми, які є на вашому пристрої та доступні, натиснувши піктограму програми. Такі програми можна знайти в магазинах доповнень таких як Play Market на Android, App Store на iOS тощо. Вони розроблені для певної операційної платформи та можуть застосовувати всі потенційні засоби пристрою. Вони також можуть розрізняти жести (стандартні жести, встановлені операційною системою, або нові жести, які використовуються в певній програмі). Локальні програми можуть отримати доступ до сповіщень пристрою, а також підтримувати функціонування в автономному режимі [3].

Такі програми шифруються тією ж мовою, що й сам мобільний пристрій. Наприклад, мова Objective C використовується для iOS і Java для операційної

системи Android. Програми мовою пристрою гарантує швидке виконання та достойний рівень безпеки.

Однак цей тип програми має найбільшу вартість, оскільки він підключений до однієї моделі ОС. Альтернативою цього є написання оновленої версії коду цієї програми для кожної платформи відповідною мовою. Це подвоює вартість. До речі більшість мобільних ігор написані мовами, що відповідають операційній системі.

Мобільні веб-додатки у дійсності не являються такими. Це веб-сайти, які багато в чому схожі на нативні доповнення, але досі не зуміли остаточно їх витіснити. Вони стартують з підтримкою веб-переглядача і зазвичай реалізовані з допомогою HTML5. При старту мобільних інтернет-додатків юзер здійснює всі дії, які відбуваються при вході на сайт, таким же чином має ймовірність «розмістити» їх на робочий стіл, організувавши сторінку закладки сайту.

Веб-додатки стали дуже популярними в той час, коли вони почали розробляти HTML5, і люди зрозуміли, що можуть отримати доступ до багатьох функцій різних програм, зайшовши на веб-сайт через звичайний браузер. Сьогодні важко точно сказати, де проходить чітка межа між веб-додатками та звичайними веб-сторінками, оскільки функції HTML5 з кожним днем збільшуються, і все більше і більше сайтів використовують HTML5.

Такі програми працюють через браузер і не потребують завантаження. Вони відрізняються від рядових сайтів тим, що сервер має бути доступний щоразу, коли клієнт здійснює дію. Найкращим прикладом веб-програми є Google Пошта.

Кросплатформенні програми — це однозначне співіснування рідних і веб-додатків. Як і локальні програми, їх можна завантажити з магазину додатків, а самі кросплатформенні додатки можуть скористатися перевагами багатьох функцій пристрою. Як і веб-додатки, їх платформа заснована на HTML5. Їх опрацьовує браузер, в самій програмі.

Часто, крім існуючих веб-сайтів, виходять такі кросплатформенні додатки як шар, щоб їх використання можна було помістити в список магазинів програмного забезпечення. Це допомагає розробляти індивідуальні мобільні додатки, не витрачаючи багато коштів в найкоротші

терміни [4].

Отже кросплатформенні доповнення такі популярні тому, що код в них написаний на одній мові програмування. Цим самим виробництво гібридних додатків дозволяє їм чудово працювати на різних операційних системах.

Вони встановлені на телефоні, але написані за допомогою веб-технологій (HTML5, CSS і JavaScript). Перш за все, це дозволяє отримати доступ до функцій телефону, які недоступні в мобільних веб-програмах. Мінусом таких програм являється дизайн, так як різні операційні системи мають різні стандарти UI/UX. Прикладом цього є, дизайн iPhone і Android.

Мобільні сайти — не явна категорія, але заслуговує на увагу. Мобільні сайти дають можливість без істотних витрат протестувати ринок, затвердити ідею або побудувати MVP. Тим самим зробити бізнес більш доступним на мобільних пристроях.

Використовуючи адаптивні CSS-фреймворки, такі як Bootstrap, можна створювати веб-додаток або веб-сайт, який адаптуються під різні телефони.

Користувачі будуть використовувати один і той же додаток / веб-сайт, але дизайн зміниться відповідно до дозволу екрану, який вони використовують.

Мобільні сайти не є чіткою категорією, але їм також потрібно приділити увагу. Вони дозволяють безкоштовно проаналізувати ринок. Це робить виробництво доступнішим на портативних пристроях. Застосовуючи адаптивні фреймворки CSS, такі як Bootstrap, ви можете створити веб-додаток або веб-сайт, сумісні з різними телефонами. Користувачі використовуватимуть ту саму програму/веб-сайт, але дизайн буде використовуватися відповідно до роздільної здатності екрана.

1.3 Аналіз існуючих рішень для розробки кросплатформенних додатків

При роботі над кросплатформенними додатками є можливість проектам виконуватись на різних платформах, включаючи iOS, Android та інші, завдяки формуванню вихідного коду програми. У такого методу є певні плюси, а саме швидкість освоєння технології та всього процесу розробки в цілому. Досягається

вона саме за можливості вдруге використовувати текст програми. Завдяки цьому розробники можуть використовувати один текст програми на декількох платформах, а це, авжеж, значно зменшує не тільки час розробки, а й грошову вартість, на відміну від розроблення нативних додатків. Якщо точніше, то можливість вдруге використовувати код програми на декількох операційних платформах допомагає знизити кошти на ресурси розробки до 50-80%[5].

Завдяки тому, що крос-платформні програми не потребують необхідності вивчати одразу кілька технологій, затрати часу також зменшуються. Так як потреби і у створенні різноманітних баз коду програми немає, то й вихідне розгортання на цільових платформах обходиться зі значною економією часу. Можливі зміни, крім цього, можна здійснювати синхронно, навіть на не вносячи на кожній операційній платформі конкретних змін. Також, черговою перевагою, є те що ніяких ускладнень з синхронізацією на різних ОС не виникає. Це може допомогти швидше та простіше збільшити кількість платформ та пристроїв, а також залучити значно більшу кількість користувачів. Але у гібридного підходу, на жаль, є певні вади. Так як у кожної платформи для взаємодій із системними сервісами є свої API, не є можливим написання коду , що буде робити абсолютно на всіх системах. Через те що у коді програми розробнику потрібно враховувати на яких у даний момент платформах запущено програму. Доволі часто, по цій причині, задля технологій створюється певна екосистема. Вона складається з обгортки на кожній платформі та особливого середовища виконання клієнтського додатку. Для користувача додатку та системними API, така обгортка стає проміжною ланкою. По цій причині можуть виникати проблеми з продуктивністю роботи додатку та швидкістю відгуку на дії користувача.

Кожна платформа може мати певні специфічні складники інтерфейсу клієнта, взаємозв’язок з якими може розходитись. Але є недолік, яка полягає в тому, що представлення знайомої манери клієнта займає надлишковий час і спрямовує код програми на платформу. Це дещо не сходиться з принципом кросплатформного підходу до розробки клієнтських додатків. Але в той же час, вживаючи особливі компоненти користувача можна критично позначитись на UX. Розробка кросплатформенних додатків може здійснюватися за допомогою різних технологій. Це можуть бути мови програмування C або C#, веб-технології, і так

далі.

В даній роботі було вирішено проаналізувати рішення та методи, які використовують веб-технології, адже вони можуть працювати на будь-якій платформі: персональні комп’ютери, телефони, планшети, автомобілі, холодильники тощо. Причина цього в тому, що засновані вони на веб-стандартах і загальних API, що використовують на цих платформах.

Ми розглянемо чотири фреймворки — PhoneGap, Ionic, Flutter, React Native і з'ясуємо, в яких випадках їх застосування буде корисно.

Тому перевагою більшості рішень є вдруге використання коду програми для запуску на різних платформах. Також однією з плюсів цієї мови розробки є JavaScript, оскільки її легко вивчити, а з жовтня 2021 року вона стане найпопулярнішою серед розробників.

Мінусами багатьох із цих результатів є необхідність роботи в екосистемі та розширення упаковки.

1.3.1 PhoneGap

PhoneGap дає можливість для використовування веб-технологій при розробці мобільних додатків. У PhoneGap використовують JavaScript разом із HTML та CSS[6]. Яскраві приклади розробок через PhoneGap це LogitechSqueezeboxController, Localeur, Untappd, HealthTap.

Для того щоб повисити комфортність кроссплатформенної розробки та подальшого тестування є можливість використання Adobe Dreamweaver версії вище ніж 5.5, MyEclipse 2013, Tiggzi, Application Craft[7]. При цьому розробка на JavaScript не викликає труднощів, особливо при умові наявності досвіду при роботі з ним.

Розроблений через PhoneGap додаток є набором HTML-сторінок в нативній оболонку. Ці сторінки можуть зберігати локально або у хмарі, а коли додаток запущений на смартфоні, то через певні плагіни вони отримують доступ до пристрою. По цій причині додатки PhoneGap важать небагато, тим не менш, в цей же час вони виглядають більш синтетично, а від особливостей конкретних ОС якість UI може бути різною.

PhoneGap не може відрізнитись високопродуктивністю, особливо при порівнянні в порівнянні з нативними інструментами. Але у кожного інструменту є свої плюси і мінуси, тому приймати рішення який використовувати потрібно з урахуванням завдання, яке цей інструмент має вирішити[8].

1.3.2 Ionic Framework / Capacitor

В 2012 році веб-технології майже не використовувались як засобу для створення власних додатків. Саме тоді було створено Ionic, основною ідеєю якого була розробка кращого способу для веб-розробників розробляти додатки для смартфонів[9].

Ionic Framework являється безкоштовним та відкритим розробкою, яка була випущена за ліцензією MIT. Іншими словами його використання в особистих або комерційних проектах є безкоштовним. Ліцензія MIT також використовується в jQuery і Ruby on Rails.

Ionic Framework по суті являється інструментарієм інтерфейсу задля розробки мобільних і десктопних додатків за допомогою таких web-технологій як HTML, CSS і JS.

Функціонал Ionic Framework спрямований на User Interface або взаємодію із UI програми. Окрім використання окремо є можливість також нескладної інтеграції з іншими бібліотеками, наприклад Angular.

Частіше за все Ionic використовується для створення додатку для розповсюдження через AppStore і PlayMarket. WebView в наборах для розробки можуть відображати будь який додаток Ionic і надалі можуть дозволяти отримувати доступ до SDK.

Capacitor — Технологія для кросплатформенної роботи додатків, яка може дозволити розробку додатків, що працюють і на iOS, і Android, і на Electron та також веб називається Capacitor.

Capacitor надає послідовний веб-API, який дає можливість додаткам найбільше наближатися до веб-стандартів. При цьому мати доступ до більшості функцій початкового девайсу на тих платформах, що їх підтримують. Щоб додати «рідний» функціонал можна легко скористатися допомогою Plugin API для Swift на iOS, Java на Android та JavaScript для Web.

1.3.3 Flutter

Flutter — це програмний пакет SDK для створення додатків для iOS, Android та Web з однією кодовою базою. Ідея Flutter в тому щоб у розробників була можливість для розробки програм однаково ефективних на різних платформах. Мовою розробки э Dart[10].

У Flutter, так само як і у React Native, використовуються компоненти для відображення в реактивному стилі. Однак, на відміно від React Native, у Flutter збирається шлях до нативного коду програми. Flutter може контролювати все що малюється на екрані і саме тому може уникати проблем із продуктивністю, через необхідність мосту JavaScript. Dart має такі характеристики:

— може забезпечувати відкритий код, масштабовану мову для розробидодатків, веб, або навіть серверів;

— може забезпечувати об'єктно-орієнтовану мову успадкування,використовуючи C-подібний синтаксис, що AOT-компілюється у рідну мову;

— може бути переведена в JavaScript;

— може підтримувати інтерфейси та абстрактні класи.

Майже всі компоненти, які створені у User Interface відбуваються за участю об'єктів перегляду, які є прикладами класу UIView. Вони також можуть виконувати роль певних контейнерів для інших класів UIView, що врешті решт формують макет. Віджети - це найближчий еквівалент UIView у Flutter. Віджети напряму не можуть не відображати компоненти iOS, але їх можна вважати способом оголошення та створення UI. Щоправда, віджети дещо відрізняються від UIView. Як мінімум у віджетів різні тривалість, адже вони не змінюються та можуть існувати лише доти поки їх не необхідно змінити. Коли сам віджет або його стан змінюється, кожного разу Flutter автоматично створює нове дерево екземплярів. Якщо порівнювати з iOS то там перегляд не відображається під час внесення змін. Там це здебільшого змінюваний об'єкт, який відображається один раз і не змінюється доки не буде об’явлений недійсний через setNeedsDisplay().

Також віджети Flutter через свою незмінність важать зовсім небагато, порівняно з UIView. Самі віджети не переглядають і не відображають нічого, а являються лише

описом UI, а також його сематики, яка, в свою чергу, перетворюється пізніше на об'єкти.

Flutter має бібліотеку Material Components. Це набір віджетів, що можуть реалізовувати вказівки дизайну матеріалів. Material Design вважається дуже гнучкою системою дизайну, вона має чудову оптимізацію для будь яких платформ, в тому числі і для iOS.

Окрім цього Flutter має в собі сучасний фрейм у стилі реагування, 2d рендеринг, готові віджети та засоби розробки. Всі ці компоненти можуть співпрацювати сумісно з ціллю створення, тестування та настройки розробки. Простими словами основну концепцію можна назвати: «Все є віджетом».

Так як віджети являються основними складовими для UI, то й кожен віджет — це незмінна його частина. Але, тим не менш, на відміну від розділення представлення в будь яких інших фреймворках, а також контролерів для перегляду, макетів, тощо, у Flutter є послідовна модель об'єкта, а саме — віджет.

Самі віджети, в свою чергу, можуть формувати ієрархію на основі композиції. Кожен віджет знаходиться в середині, має можливість наслідування властивостей від батьківського віджету. Навіть замість об’єкта «додаток», використовується віджет, який цю роль може виконувати.

1.3.4 React Native

React Native — це фреймворк на JavaScript необхідний для розробки мобільних додатків на платформах iOS та Android. Базується він, як можна здогадатись, на React - бібліотеці JS Facebook яка використовувалась для написання UI, але орієнтований він не на браузер, а на мобільні платформи. Тобто розробники мають можливість створювати додатки, що можуть які виглядають повністю нативними. Окрім цього, React Native значно спрощує написання і для Android і для iOS, по тій причині, що майже всі коди можна розділити між платформами.

Розробки React Native записуються за допомогою JavaScript та XML-розмітки (JSX), подібним чином як у випадку React для Веб. Після цього React Native може скористатись вбудованими API візуалізації, а саме Objective-C (iOS) або Java (Android). По цій причині додаток має змогу відображатись через реальних

компонентів UI, а не веб-перегляду і саме тому він буде мати вигляд як у звичайного мобільного додатку. Окрім цього React Native може відкривати інтерфейси JavaScript для API платформ, через що розробки React Native будуть мати можливість отримувати доступ до камери, GPS, або іншим функціям платформи[11].

Наразі React Native підтримує лише iOS та Android, але, скоріше за все, він матиме можливість поширитись і на інші платформи. Майже весь код написаний на React Native може бути повністю кросплатформним. Facebook, Palantir і TaskRabbit та багато інших компаній вже активно користуються React Native для розробки додатків націлених на юзерів.

Той факт, що React Native фактично відображає за допомогою стандартних API візуалізації своєї хост-платформи, дозволяє йому виділятися з більшості існуючих методів розвитку платформних додатків, наприклад Cordova чи Ionic. Такі способи створення телефонних програм за допомогою комбінування Java, HTML і CSS в основному відображаються за підтримки веб перегляду. Хоч і підхід такого типу може працювати, він також має «мінуси», в тому числі відносно ефективної роботи. Також, вони зазвичай не мають можливості отримати доступ до власних частин інтерфейсу хоста. В той час, коли ці рамки пробують повторити природні частини інтерфейсу, результати зазвичай «відчуваються» лише трохи; реверсивний інжиніринг всіх тонких речей, таких як анімування, вимагає величезних зусиль, і вони можуть швидко зустрітися.

Протилежно від цього, React Native фактично переводить вашу розмітку на реально існуючі, рівні частини інтерфейсу, за допомогою використовування існуючих засобів візуалізації поглядів на любі платформі, що задіяна в вашій праці. Крім того, React працює окремо від основного потоку інтерфейсу, тому ваша програма може підтримувати високу продуктивність без шкоди для можливості. Цикл оновлення в React Native такий же, як і в React: коли параметри або стан змінюються, React Native повторно перемальовує інтерфейс. Основна відмінність React Native від React у веб-переглядачі полягає в тому, що React Native це робить, використовуючи бібліотеки інтерфейсу користувача на своїй хост-платформі, а не використовуючи розмітку HTML та CSS.

Для розробників, які звикли працювати в Інтернеті з React, це означає, що вони мають можливість писати мобільні програми з продуктивністю та виглядом власної програми, використовуючи при цьому звичні інструменти. React Native також означає покращення порівняно з нормальним мобільним сімнадцятим розвиненням у двох других сферах: уміння розробника та динаміка розвинення платформ.

Робота з React Native може швидко зменшити ресурси, необхідні для створення мобільних додатків. Будь який розробник, який має навички писати текст програми за допомогою React, може зробити орієнтацію на Інтернет, iOS та Android, і все з тим самим набором навиків. Забираючи критерій необхідності синхронізування розроблювальників на платформах різного цільового призначення, React Native дає можливість вашій команді швидше створювати зміни та тим ефективніше ділитися навичками та ресурсами.

Окрім колективних знань, теж можна передати велику частину програмного тексту. Не весь об’єм тексту програми, яка створюється, буде кросплатформним, через це в залежності від того, яка функція потрібна на певній платформі, розробникам може на деяких проміжках часу знадобиться використовувати Objective-C або Java. Але повторно використовувати текст додатку на платформах дуже легко з React Native. Наприклад, програма Facebook Ads Manager для Android поділяє 87 відсотків своєї кодової бази з версією iOS, як зазначено в програмі React Europe 2015.

Як і в усьому, використовування React Native не позбавлене й мінусів, і те, чи дійсно використовування React Native як технології буде ефективнішим, залежить від індивідуальної ситуації.

Найбільший ризик — це, можливо, зрілість React Native, тому що проект ще відносно молодий. Підтримка iOS була розпочата в березні 2015-го року, а підтримка Android була розпочата у вересні 2015-го року. Документація безпосередньо має можливість покращити і продовжує розвиток. Деякі функції на iOS та Android все ще не підтримуються, і угрупування все ще має на меті знайти найкращі практики. Перевагою є те, що в переважній більшості випадків доступна можливість самостійно реалізувати підтримку відсутніх API.

1.4 Аналіз системи керування бази даних.

Для реалізації бази даних застосовують такі засоби, як MySQL, Oracle або SQL Server, щоб розкривати, захистити або відредагувати дані та повернути їх клієнту за підтримкою коду візуальної складової. Раніше згадані мови застосовуються не тільки для створення веб сайтів, програмного забезпечення та додатків; їх також застосовують для формування та керування СУБД. Використання баз даних є неможливим без відповідних для них мов відмінних від тих на якій програмується код програми, тобто стандартну, для доступу та обробки реляційних баз даних наприклад SQL, що розшифровується як Structured Query Language. В неї є приватна розмітка, що дозволяє розробникам працювати і зберігати дані в системі. Дослідимо різні типи, плюси та мінуси баз даних більш детально (табл. 1.1).

Таблиця 1.1— Бази даних

№	Назва	Переваги	Недоліки
1	Oracle	Вона має інновації та нові функції у своїх продуктах, оскільки Oracle намагається встановити планку для інших інструментів керування базами даних. Вони також неймовірно надійні, і ви можете знайти той, який може робити майже все, що ви думаєте.	Можуть знадобитися значні ресурси, тому реалізація Oracle може вимагати оновлення обладнання. Ви можете витратити багато часу та зусиль, щоб змусити MySQL виконувати те, що інші системи роблять автоматично.
2	MySQL	Безкоштовний. Пропонує багато функцій. Це можна зробити для роботи інші бази даних, в т.ч DB2 і Oracle.	Ви повинні спробувати змусити MySQL виконувати те, що інші системи роблять автоматично. Підтримка вбудованого XML або OLAP недоступна Підтримка доступна для безкоштовної версії, але за неї доведеться платити.

3	PostgreS QL	Ця система управління базою даних є масштабованою і може обробляти терабайти даних. Підтримує JSON. Є кілька попередньо визначених завдань. Існує ряд інтерфейсів.	Чіткого документа немає. Конфігурація може бути заплутаною. На швидкість можуть вплинути великі масові транзакції або запити.

Продовження таблиці 1.1 — Бази даних

4	Microsoft SQL Server	Це дуже швидко і стабільно. Рівні продуктивності можна регулювати та контролювати, що зменшує використання ресурсів. Візуалізація доступна на мобільних пристроях. Дуже добре працює з іншими продуктами Microsoft.	Ціни для підприємств можуть бути вищими, ніж для багатьох організацій. Навіть якщо ви налаштуєте продуктивність, Microsoft SQL Server може керувати ресурсами. Багато людей мають проблеми з використанням інтегрованих служб SQL Server для імпорту файлів
5	MongoDB	Швидкісна і легка у використанні. Підтримує JSON та інші документи NoSQL. Легко отримати доступ до її структури даних. SQL не використовується як мова запитів.	Установки за замовчуванням не захищені Налаштування може бути тривалим процесом. Інструменти для перекладу SQL на запити MongoDB доступні, але вони додають додатковий крок до використання системи.
6	MariaDB	Шифрування доступне на мережевому, серверному та додатковому рівнях. Розширювана архітектура та плагіни дозволяють налаштувати все за потребами. В системі велика швидкодія і стабільність.	Як і у багатьох інших безкоштовних двигунів бази даних,підтримка не безкоштовна. Вона все ще нова, тому немає впевненості, що наступні оновлення та версії будуть найближчими.

Під час реалізації цього проекту альтернатива пала на сторону MySQL. По-перше, ця база даних не потребує додаткових коштів. По-друге, інтегрований продукт MySQL Workbench дає можливість спілкуватися тільки з встановленою базою даних, що дає перспективу виконувати будь-яку обробку даних через дружній інтерфейс. Ця база даних є однією з найпопулярніших у світі за

популярністю і поступається лише корпоративній базі даних Oracle. Це свідчить про те, що в разі проблем або багів у роботі цієї системи мережа може відшукати велику кількість матеріалу для їх врегулювання.

2 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНО-ПОШУКОВОГО ДОДАТКУ

2.1 Модель використання додатку

За підсумками опрацювання проекту було виділено цільову аудиторію майбутнього додатку (табл.2.1) та утворенно опис варіантів (рис.2.1) його застосування (табл.2.2).

Таблиця 2.1 — Розподіл функцій користувачів

Роль	Опис дій
Клієнт	Користувач додатка, що використовує функції «гість».
Лікар	Користувач додатка, що використовує функції «лікар».
Адміністратор	Користувач додатка, що використовує функції «адміністратора».

Таблиця 2.2 — Опис варіантів використання

Назва дії	Опис
Вхід в систему	Функція входу в систему.
Реєстрація в системі	Функція, щоб надати можливість користувачу зарегіструватись як “гість ” або “лікар”.
Залишити коментар	Функція, щоб надати можливість користувачу залишити коментар.
Перегляд інформації	Кожен користувач має можливість переглянути інформацію в додатку.
Пошук інформації	Можливість виконувати пошук інформації в додатку.
Редагувати дані	Адміністратор може редагувати дані в додатку.
Додавати публікації	Адміністратор може додати нові публікації.

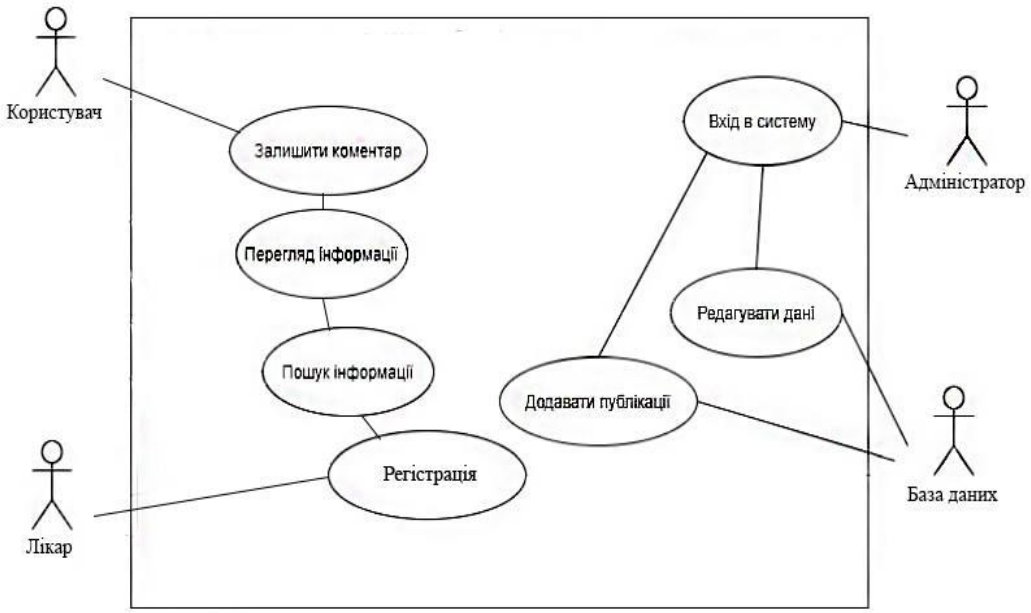


Рисунок 2.1 — Діаграма варіантів використання

Тепер сформуємо послідовність входу користувача в систему, та його там авторизацію (рис.2.2).

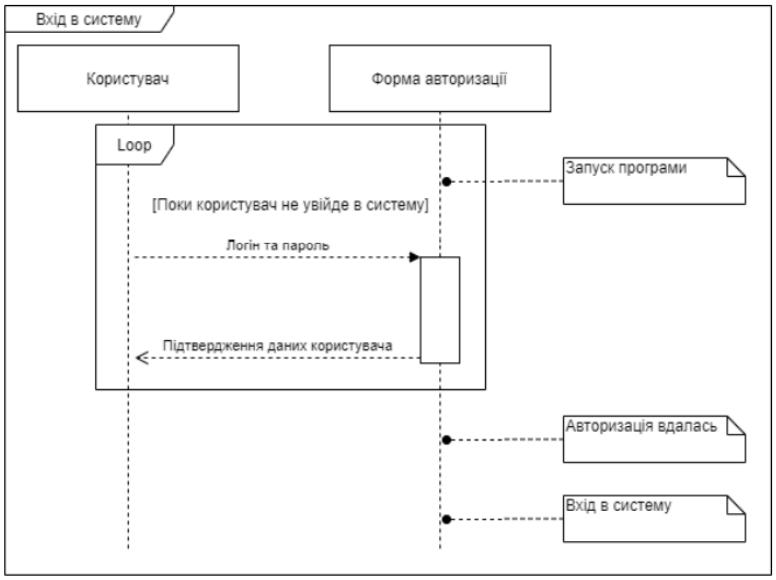


Рисунок 2.2 — Діаграма послідовності входу в систему

Розглянемо діаграму методики послідовності коментування в програмі(рисунок 2.3).

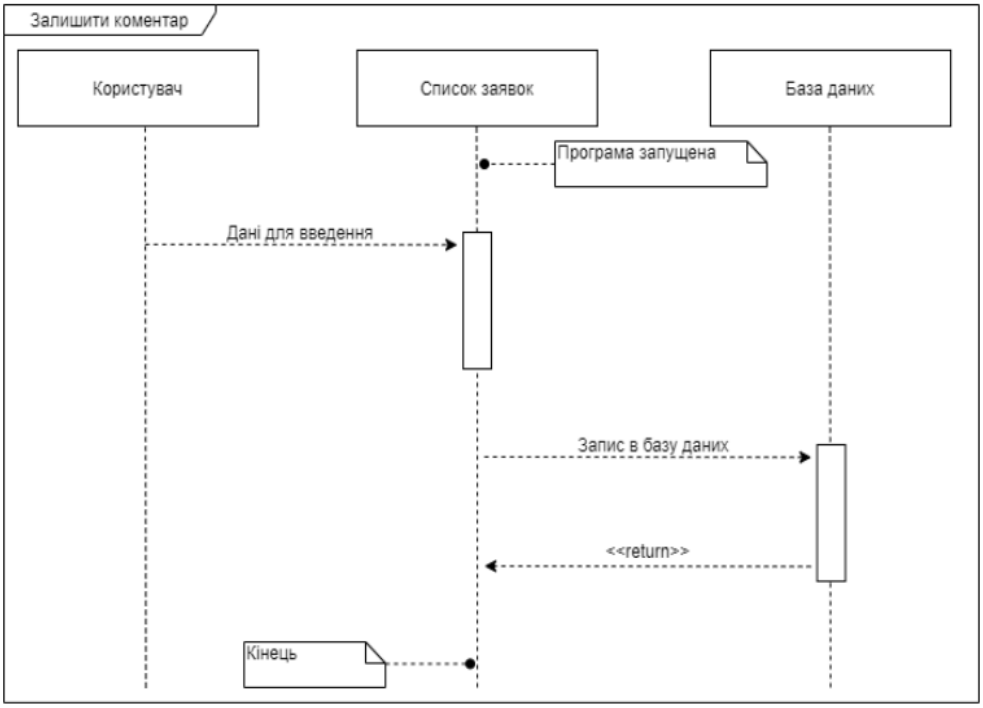


Рисунок 2.3 — Діаграма послідовності коментування

Також було розроблено діаграма пошуку та перегляду інформації в самому додатку(рисунок 2.4)

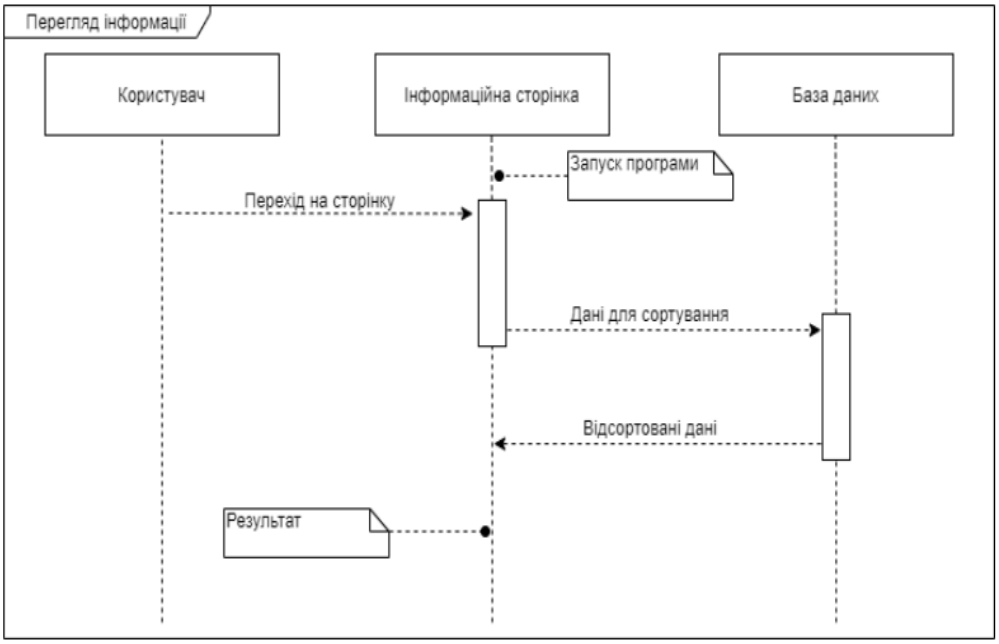


Рисунок 2.4 — Діаграма послідовності перегляду інформації

На наступному зображені (рисунок 2.5) наведена діаграма запису та збереження інформації в базу даних.

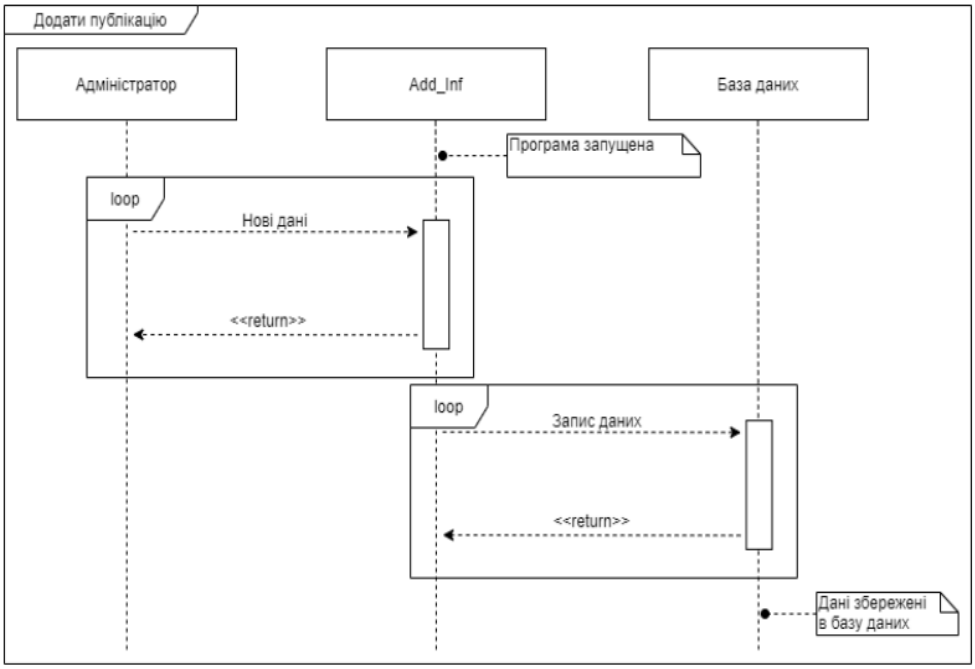


Рисунок 2.5 — Діаграма послідовності додавання інформації

На зображені (рисунок 2.6) наведена діаграма послідовності пошуку та сортування в базі даних потрібної для клієнта інформації.

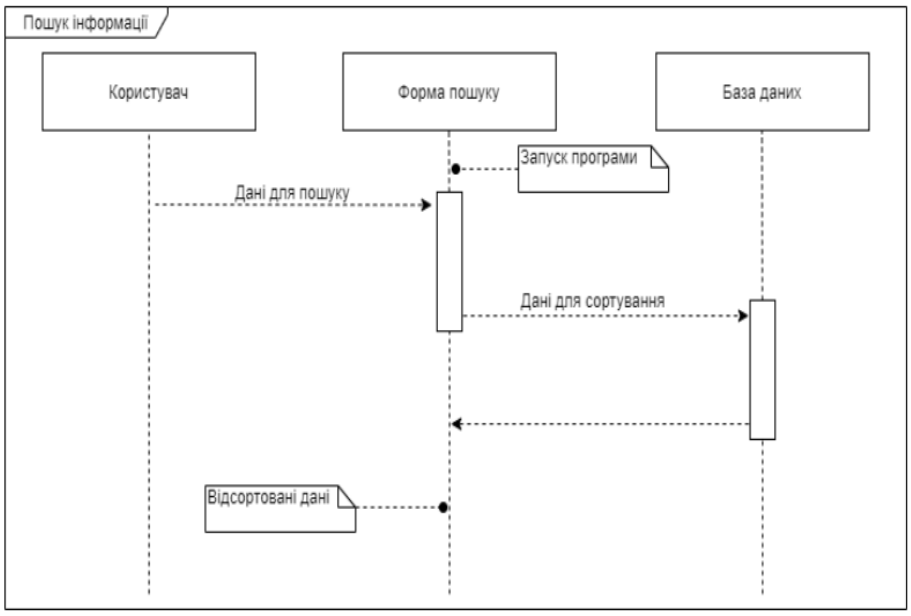
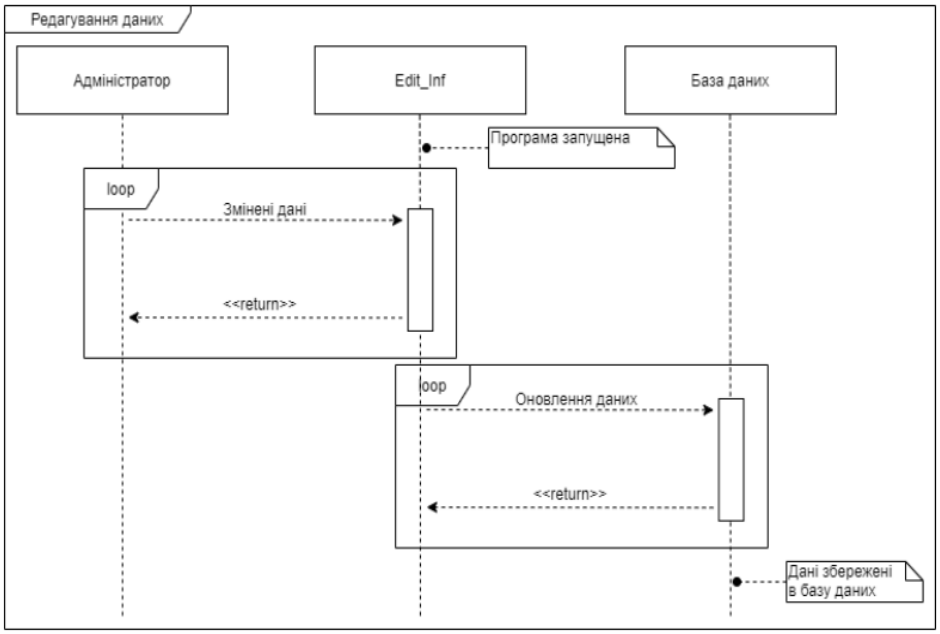


Рисунок 2.6 — Діаграма послідовності пошуку інформації

На наступному зображені ми можемо візуалізувати процедуру послідовності редагування інформації в кросплатформенній програмі (рисунок 2.7).



2.2 Проектування пошуково-інформаційної системи

Прикладом дизайну є проста діаграма сторінки (рисунок 2.8), на якій у вигляді ескізу або html-документа відображаються структурні елементи майбутньої програми: меню, кнопки, таблиці тощо. Прототипом (рисунок 2.9) може бути статичне зображення або динамічний html-документ.

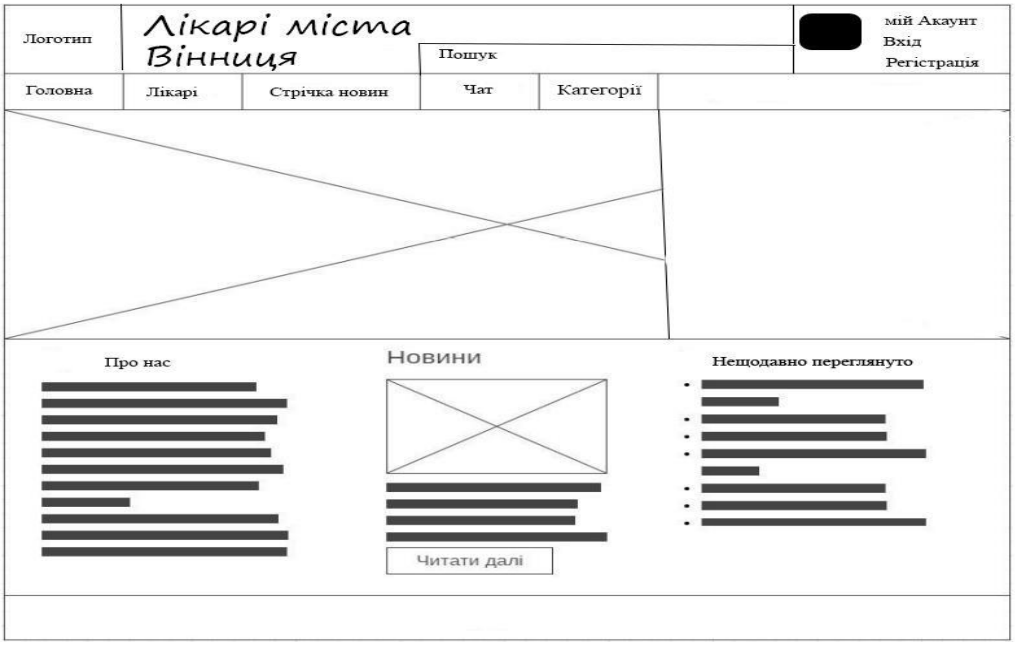


Рисунок 2.8 — Прототип дизайну головної сторінки

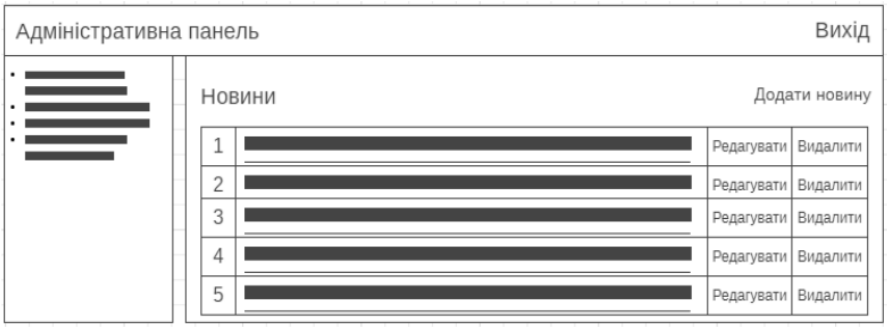


Рисунок 2.9 — Прототип дизайну адміністративної панелі

2.3 Проектування моделі бази даних

При створенні бази даних потрібно контролювати процедуру, що зберігається в ній, щоб в ході наступної роботи інформацію можна було легко отримати або змінити. Для цього потрібно мати структуру вхідних даних. Структурування — це набір домовленостей про те, як подається інформація. Зрозуміло, що інформацію можна конструювати різними способами. Залежно від структури розрізняють ієрархічні, мережеві, реляційні, об'єктно-орієнтовані та гібридні моделі баз даних. Сьогодні це найпопулярніша структура відносин.

Вхідні дані — це набір даних, необхідних для належної роботи системи. Аналіз необхідних імпортованих даних полягає у формуванні вимог, які пропонуються для нових або модифікованих матеріалів, з урахуванням потенційно суперечливих вимог різних замовників. На основі вивчення був проведений розбір, в підсумку якого було розроблено схему даних у програмі MySQL (рисунок 2.10).

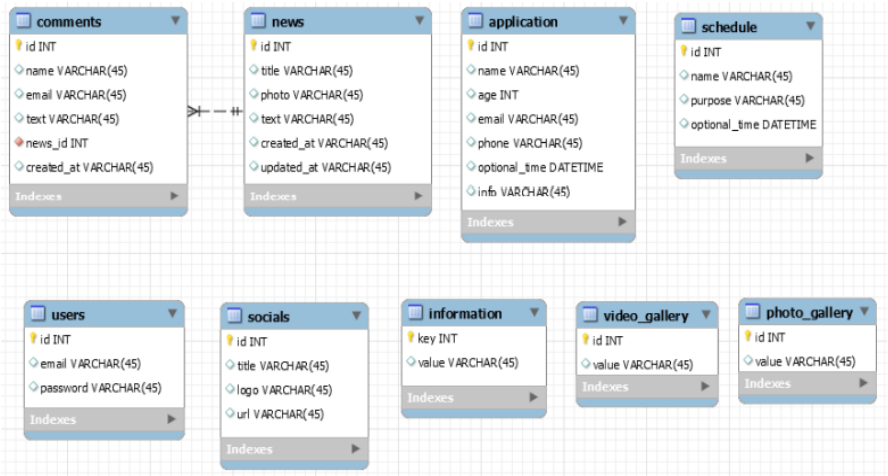


Рисунок 2.10 — Схема даних, використовуваних у додатку

Базу даних було створено в програмі MySQL, що нараховує 9 таблиць. Таблиця «Новини» (рисунок 2.11) включає замітки про новини, оприлюднені на сайті. Вона містить чотири стрічки, де розміщені: новини заголовка, дані, фотографії, інформація по вводу та поновленню.

#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно	Действие
1	id	bigint(20)		UNSIGNED	Нет	Нет		AUTO_INCREMENT	
2	title	varchar(255)	utf8mb4_unicode_ci		Нет	Нет			
3	photo	varchar(255)	utf8mb4_unicode_ci		Да	NULL			
4	text	text	utf8mb4_unicode_ci		Нет	Нет			
5	created_at	timestamp			Да	NULL			
6	updated_at	timestamp			Да	NULL			

Рисунок 2.11 — Таблиця «news»

В таблиці коментарів (рисунок 2.12) є інформація про коментарі, які користувачі залишили в додатку. Таблиця містить 6 стрічок, де знаходяться ключ, ім’я користувача, ip, текст, ідентифікатор і дата написання.

#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно	Действие
1	id	bigint(20)		UNSIGNED	Нет	Нет		AUTO_INCREMENT	
2	name	varchar(255)	utf8mb4_unicode_ci		Нет	Нет			
3	email	varchar(255)	utf8mb4_unicode_ci		Нет	Нет			
4	text	text	utf8mb4_unicode_ci		Нет	Нет			
5	news_id	bigint(20)		UNSIGNED	Нет	Нет			
6	created_at	timestamp			Да	NULL			

Рисунок 2.12 — Таблиця коментарів

В таблиці публікацій (рис. 2.13), в якій знаходяться дані клієнта, а також дод. інформація. В таблиці знаходяться сім стрічок які складаються з ключа, ім’я користувача, та email.

#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно	Действие
1	id	bigint(20)		UNSIGNED	Нет	Нет		AUTO_INCREMENT	
2	name	varchar(255)	utf8mb4_unicode_ci		Нет	Нет			
3	age	int(11)			Нет	Нет			
4	email	varchar(255)	utf8mb4_unicode_ci		Нет	Нет			
5	phone	varchar(255)	utf8mb4_unicode_ci		Нет	Нет			
6	optional_time	varchar(255)	utf8mb4_unicode_ci		Нет	Нет			
7	info	text	utf8mb4_unicode_ci		Да	NULL			

Рисунок 2.13 — Таблиця публікацій

В таблиці юзерів (рис. 2.14) знаходиться інформація про них в додатку. Таблиця містить чотири стрічки які складаються з ключа, email, пароль, токену для того щоб зберегти інформацію про авторизацію.

#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно	Действие
1	id	bigint(20)		UNSIGNED	Нет	*Нет*		AUTO_INCREMENT	
2	email	varchar(255)	utf8mb4_unicode_ci		Нет	*Нет*			
3	password	varchar(255)	utf8mb4_unicode_ci		Нет	*Нет*			
4	remember_token	varchar(100)	utf8mb4_unicode_ci		Да	*NULL*			

Рисунок 2.14 — Таблиця користувачів

3 РОЗРОБКА КРОСПЛАТФОРМЕНОГО МОБІЛЬНОГО ДОДАТКУ

3.1 Розробка додатку на фреймворку Expo

Javascript — це динамічна, об'єктно-орієнтована мова програмування для створення прототипів. Вона може бути задіяна як на серверній частині (express js, koa js), так і на клієнтській. JavaScript класифікується як макет (під об'єктно-орієнтованим комплектом), мова програмування скриптів з динамічним типом. Вона не тільки створює прототипи, JavaScript також певною мірою підтримує інші шаблони розробки (імперативні та відносно функціональні) та деякі суміжні архітектурні особливості, : динамічне та слабке введення, автоматичне керування пам’яттю, успадкування прототипів та функції першокласних об’єктів.

React — доступна JavaScript-бібліотека для написання інтерфейсу користувачів, яка створена вирішувати проблематику частинного обновлення інформаційності веб сторінки, з якою зустрічаються в розробленні застосунків на одну сторінку. React дає змогу розроблювальникам створювати масштабні веб застосунки, які можуть використовувати дані, котрі з часом мають можливість змінюватись, без оновлення сторінки. Його ціль лежить в тому, щоб бути швидким, масштабованим, а також простим.

Фреймворк Expo це платформа для універсальних програм, заснованих на JavaScript-бібліотеці React. Цей фреймворк складається з набору готових інструментів та сервісів на основі React Native та native платформ, які допомагають розробляти, створювати, розгортати та швидко виконувати ітерації на iOS, Android та веб-додатках з однієї і тієї ж кодової бази JavaScript/TypeScript. У даному фреймворку існує два основних підходи до створення програм керований (managed) і простий (bare) підхід. За допомогою керованого підходу пишуться тільки інструменти та сервіси JavaScript/TypeScript, а все інше фреймворк Expo зробить сам.

Отже, було досліджено інноваційний на сьогодні інструмент для розробки додатку для всіх платформ та його ключові елементи. Проаналізувавши всі платформи було встановлено, що на сьогодні нативні додатки мають перевагу у продуктивності, проте швидкість розробки і універсальність коду має Expo. Я вважаю що у майбутньому продуктивність цих додатків буде однакова та процес

створення додатків буде швидший. Досліджені всі переваги Expo. Встановлено, що для створення додатку для всіх платформ необхідно знати лише мову програмування Javascript в той час для розробки додатків без Expo необхідно знати як мінімум 3 мові (Kotlin/Java, Swift, Javascript). Та витратити мінімум у 3 рази більше часу та в 3 рази більше репозитаріїв для зберігання коду. Підсумовуючи можна сказати що Expo вперше дає нам можливість створювати додатки на всіх платформах і не витрачати на це багато часу.

3.2 Програмно-апаратна реалізація

Сама розробка буде відбуватись у безкоштовному текстовому редакторі VS Code. Visual Studio Code — це крос-платформний редактор скриптів, створений корпорацією Майкрософт.

Окрім цього для зберігання першого коду, а також контролю версій сайту було задіяно систему контролю версій Git. Вони дають можливість розробникам певним чином зберігати всі зміни, що були внесені в код програми. Через це, якщо виникають помилки, вони мають можливість просто відновити код до вихідного стану замість того, щоб тратити години на знаходження незначних помилок, що створюють проблеми для всього коду. Такі системи контролювання версій також позволяють декільком розробникам виконувати свою роботу над одним проектом і зберігати внесені змінювання, щоб бути впевненими в тому, що всі можуть контролювати те, над чим вони працюють.

Розроблення сайту розпочинається з генерації нового проекту за допомогою використовування фреймворку Expo. Задля цього в Composer необхідно написати наступну команду:

composer create-project --prefer-distExpo/Expo search for doctors

Де «search for doctors» — ім'я проекту і може бути змінено на будь-яке. Після чого буде створено базову архітектуру проекту (рис. 3.1).

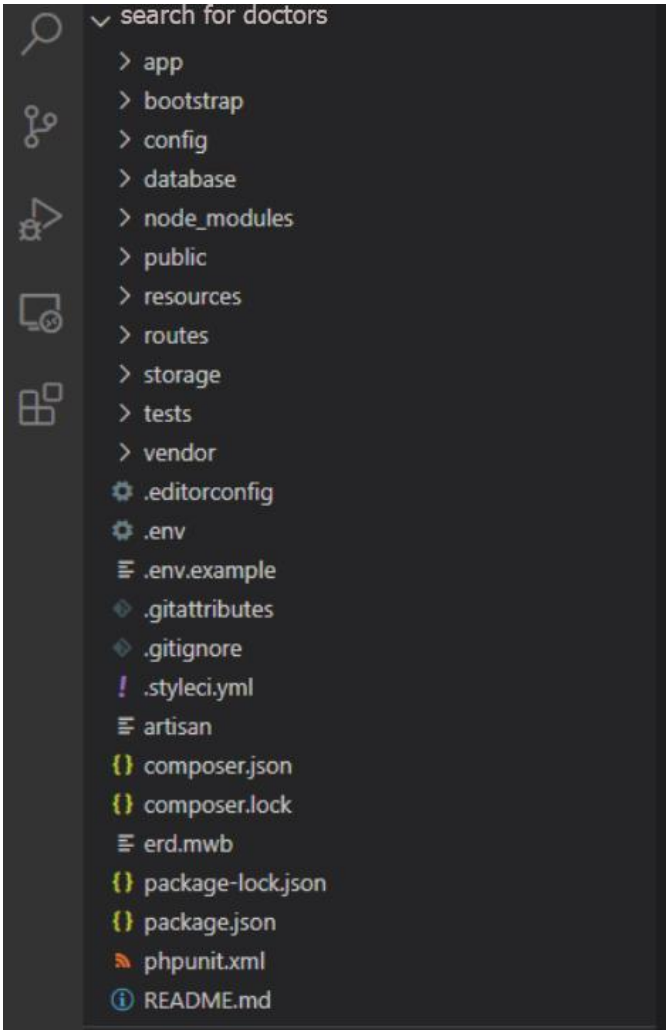


Рисунок 3.1 — Архітектура проекту

В наступному розпочинається розроблення частини сайту для клієнтів, а саме генерація всіх необхідних частин, налаштування маршрутів доступу, а також частини шаблонів, розроблення модулів частини серверу сайту, генерація всіх контролерів, що необхідні, моделей та налаштувань API роутеру.

3.2.1 Розроблення частини сайту для клієнтів

Створення сайту розпочинається з написання «скелету» за допомогою мови розмітки HTML. Після цього за допомогою мови стилю каскадів CSS створюється зовнішній вид веб-сайту і накладається візуальна форма частин для зручного використовування. Потім новостворений шаблон руйнується на окремі частини, що дає можливості багаторазового використовування одного компоненту на різних сторінках сайту. Приклад компоненту головної сторінки наведено в додатку В. Всі компоненти знаходяться в папці resources/js/.

Наступний крок має на меті створення єдиної точки входу, що підключає всі необхідні частини, а також модулі JavaScript. (рис. 3.2).

```
import router from "./router";
import Vuetify from "vuetify";
import VueSweetalert2 from 'vue-sweetalert2';
import 'sweetalert2/dist/sweetalert2.min.css';
window.Vue = require('vue');
window.Vue.use(Vuetify);
window.Vue.use(VueSweetalert2);
import AppComponent from "./components/AppComponent";
const app = new Vue({
 el: '#app',
 components: {
 AppComponent
 },
 router
});
```

Рисунок 3.2 — Створення точки підключення

Створивши всі необхідні шаблони і компоненти. Відбувається налаштування маршрутів для навігації між сторінками сайту. Далі наведено приклад коду маршруту для головної сторінки(рис.3.3).

```
export default new Router({
    mode: "history",
    routes: [
    {
    path: '/',
    name: 'home',
    component: HomeComponent
    }
    ]
  });
```

Рисунок 3.3 — Код маршруту

В результаті при кліку на посилання відбувається перехід на необхідний компонент вказаний в маршрутизаторі.

3.2.2 Розробка серверної частини

Розробка серверної частини починається зі створення та підключення бази

даних. Для цього необхідно перейти до налаштувань підключення до бази даних. Необхідно змінити файл (.env), він знаходиться в кореневій папці проекту. Необхідно змінити 4 рядки: DB_HOST = localhost, DB_DATABASE = doctor, DB_USERNAME = root, DB_PASSWORD = secret. Де DB_HOST — ім'я хоста, DB_DATABASE — ім'я бази даних, DB_USERNAME — ім'я користувача для доступу до бази даних, DB_PASSWORD — пароль користувача для доступу до бази даних.

Потім створюються міграції для того щоб автоматично було створено базу даних. Міграції — це система контролю версій для бази даних. Вони дозволяють команді програмістів змінювати структуру БД, в той же час залишаючись в курсі змін інших учасників. Далі наведено приклад міграції для створення таблиці користувачів. (рис.3.4).

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

Рисунок 3.4 — Міграція

Після генерації всіх міграцій, що є необхідними, потрібно прописати в терміналі Composer наступну команду:

php artisan:migrate

Наступним кроком є генерування моделі для комфортної роботи з базою даних, а також налаштувань залежностей між таблицями і приєднання модулів додаткового призначення. Нижче наведено приклад моделі для користувачів (рисунок 3.5).

```
class User extends Authenticatable
{
use HasApiTokens,Notifiable;
protected $fillable = [
'email', 'password',
];
protected $hidden = [
'password', 'remember_token',
];
}
```

Рисунок 3.5 — Модель користувачів

Всі запити що надходять на сервер обробляються API маршрутизатором, що знаходиться в файлі routes/api.php. Маршрутизація визначає, як додаток відповідає на клієнтський запит до конкретної адреси. Потім в залежності від типу запиту підключається необхідний контролер з необхідним методом який опрацьовує дані отримані з бази даних.

Контролер — з'єднує моделі, види і інші компоненти необхідні для роботи системи. А також відповідає за обробку запитів користувача, а саме отримання і обробку даних та відправку готового результату клієнтській частині системи. Приклад такого контролеру для отримання списку всіх новин наведено нижче:

```
class NewsController extends Controller
{
function get() {
$data = News::get();
return response()->json($data);
}
}
```

Рисунок 3.6 — Контролер новин

3.3 Тестування програмного додатку

Тестування програмного додатка — це один з важливих кроків розробки ПЗ. Метою тестування програмного додатка є отримання інформації щодо якості програмного продукту. Також тестування ПЗ запобігає низці ризиків, що виникають при використанні програмного додатка користувачем.

Тестування розробленого програмного додатка відбувалось у декілька етапів. Спочатку були створені автоматичні тести для тестування основних функцій програмного додатка для перевірки коректності роботи розроблених класів та взаємодії одне з одним. Для того, щоб провести такого роду тестування було проведено реорганізацію ділянок коду, які будуть тестуватись. Реорганізація проходила за наступними правилами:

— кожен фрагмент коду повинен мати на меті бути окремим об'єктом і він не повинен мати доступу до інших об'єктів — це допомагає уникнути заплутаності в коді;

— підтримування можливості налаштування замість того, щоб ставити конкретні значення певним параметрам;

— кожен метод будь-якого класу повинен бути простим і коротким для інтуїтивного розуміння фрагменту коду;

— для створення об'єктів класів слід використовувати конструктори — це допоможе створенню правильних копій об'єкту класу для коректного тестування.

Автоматизоване тестування відбувається з використанням технології Appium. Це інструмент для автоматизованого тестування в додатках з ОС iOS. Appium є кросплатформним, це означає, що: він дозволяє створювати автоматизовані тести на декількох платформах використовуючи один і той самий API. Розробник має можливість використати код повторно з наборами тестування для iOS, Android і Windows.

Після успіху в проведенні автоматичного тестування було застосовано метод «чорного ящику» для тестування. Була створена команда —п'яти осіб, що мали інтерес та які виявили бажання пройти тестування цього продукту. Кожен з цих людей користувався програмним додатком протягом 10 хвилин. При переході з одного екрану на інший вони заповнювали попередньо сформований лист, де треба було вказувати проблеми та враження від користування програмним продуктом. Також, їм необхідно було записати те, що вони хотіли б змінити в додатку та які функції додати.

У процесі цього типу тестування було виявлено кілька незначних помилок у створені та взаємодії інтерфейсу:

— було знайдено граматичну помилку у тексті повідомлень;

— затримка у 2 секунди при натисканні на пункт в меню «Стрічка новин».

Перша помилка була зроблена через людський фактор, яку неможливо було відслідкувати під час автоматичного тестування. Друга помилка була пов’язана з особливістю реалізації цієї функції і після коректування реалізації ця проблема зникла.

3.4 Головна сторінка додатку

Головна сторінка складається з шапки де знаходиться назва, логотипу, навігаційного меню, слайд шоу, блоку з інформацією про реєстрацію, кнопки з останніми новинами та відгуками (рис. 4.1).

Рисунок 4.1 — Головна сторінка

Вид головної сторінки змінюється залежно від пристрою на якому вона відкрита(рисунок 4.1)

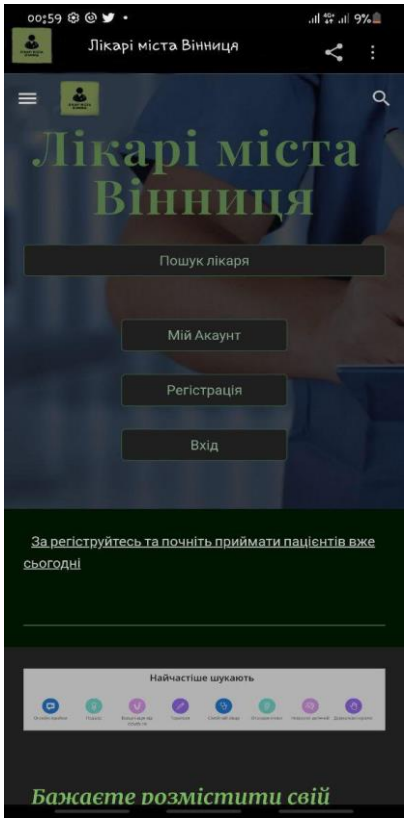


Рисунок 4.2 — Головна сторінка мобільна версія

Пункт головного меню містить в собі посилання на головну сторінку, лікарів, вибране, категорії та стрічку новин (рис.4.3).

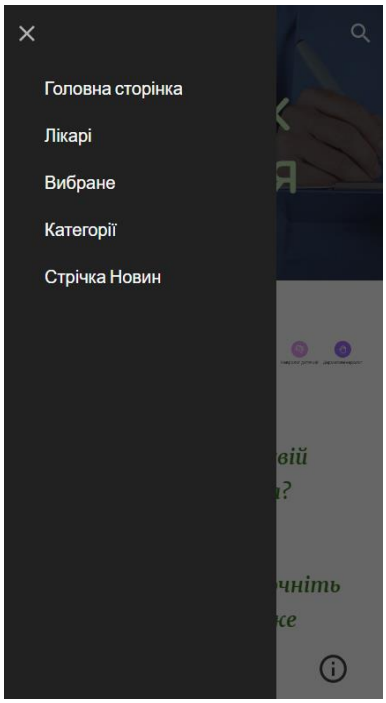


Рисунок 4.3 — Меню додатку

ВИСНОВКИ

Під час реалізації бакалаврської дипломної роботи було визначенно актуальності предметної області та проблеми, що вирішує програмний додаток, оглядженно та проаналізовано сучасні технологій для розробки кросплатформенних додатків, проаналізовано типи розробки мобільних додатків та проведений аналіз системи керування базами даних.

Спроектовано інформаційно-пошуковий додаток, спроектуванномодель використання додатку, спроектувано моделі бази даних, та сама розробка. Проведено тестування та використання програмного додатку, наведено приклад використання головної сторінки додатка.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1 List of countries by number of Internet users: [Електронний ресурс]. Режим доступу до ресурсу: https://en.wikipedia.org/wiki/List_of_countries_by_number_of_Interne t_users.

2 Дослідження: 45% усіх дорослих жителів України вже користуються смартфонами, а до 2020 року їх кількість збільшиться до 70% [інфографіка]: [Електронний ресурс]. — Режим доступу до ресурсу: https://itc.ua/news/issledovanie-45-vseh-vzroslyih-zhiteley-krainyi-uzhe-polzuyutsya-smartfonami-a-k-2020-godu-ih-proniknovenie-vozrastet-do-70-infografika.

3 Нативні додатки: [Електронний ресурс]. — 2019.— Режим доступу до ресурсу: https://ua.wikipedia.org/wiki/%D0%9D%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D1%8B%D0%B5_%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5.

4 Avivi IT Academy | Хмельницький | Які бувають типи мобільних додатків та для яких цілей вони призначені[Електронний ресурс]. — Режимдоступу до ресурсу: https://avivi.academy/blogs/tipi-mobilnikh-dodatkiv/

5 Нативна чи кросплатформенна розробка — що краще?: [Електронний ресурс]. — 2019. — Режим доступу до ресурсу: http://wnfx.ua/nativnaya-ili-krossplatformennaya-razrabotka-chto-luchshe/

6 Garbade, M. Native vs. cross-platform app development: pros and cons [Електронний ресурс] / Dr. Michael J. Garbade. Режим доступу: https://codeburst.io/native-vs-cross-platform-app-development-prosand-cons-49f397bb38ac.

7 LeRoux, B. PhoneGap Beliefs, Goals, and Philosophy [Електронний ресурс] / Brian LeRoux. Режим доступу: https://phonegap.com/blo g/2012/05/09/phonegap-beliefs-goals-and-philosophy/.

8 Trice, A. PhoneGap Explained Visually [Електронний ресурс] / Andrew Trice. ¾ Режим доступу: https://phonegap.com/blog/2012/05/ 02/phonegap-

explained-visually/.

9 Ionic, Core Concepts [Електронний ресурс] / Ionic. Режим доступу: https://ionicframework.com/docs/intro/concepts.

10 Flutter, Flutter for iOS developers [Електронний ресурс] / Flutter. ¾ Режим доступу: https://flutter.dev/docs/get-started/flutter-for/ios-devs.

11 Eisenman, B. Learning React Native [Text] / Bonnie Eisenman. 2nd edition. Sebastopol: O'Reilly Media, 2017. 242 p/

ДОДАТОК А

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
_____д.т.н., професор Азаров О.Д.
"_10_"___лютого_____ 2022__ року

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання дипломної роботи

«Кросплатформенна інформаційно-пошукова система лікарів міста Вінниці»

08-23.БДР.007.00.000 ПЗ

Студентка Колеснікова Н.С.
(підпис)

Науковий керівник Крупельницький Л.В.
(підпис)

Вінниця 2022

1 Підстава для виконання дипломної роботи (ДР)

Підставою для розробки даної бакалаврської дипломної роботи є наказ ВНТУ № від «__» ___ 2022 року та рішення засідання кафедри обчислювальної техніки (протокол №__ від «_____»__ 2022 року).

2 Мета і призначення ДР

Мета — створення кросплатформенної інформаційно-пошукової системи лікарів міста Вінниці.

Призначення розробки — допомога в пошуку потрібного медичного спеціаліста для користувачів системи

3 Вихідні дані для виконання БДР

Дані для виконання: інформація про лікарів з відкритих джерел, графічний логотип програми, задачі інформаційно-пошукової системи, вимоги до функціональних можливостей.

4 Перелік задач, що повинні бути виконані

Задачі для виконання:

— визначення актуальності предметної області та проблеми, що вирішує програмний додаток, огляд та аналіз сучасних технологій для розробки кросплатформенних додатків, аналіз типів розробки мобільних додатків та аналіз системи керування базами даних;

— проектування інформаційно-пошукового додатку, проектування моделі використання додатку та проектування моделі бази даних;

— розробка кросплатформеного додатку на фреймворку Expo, програмно-апаратна реалізація, розробка клієнтської та серверної частини;

— тестування та використання програмного додатку, використання головної сторінки додатка.

5 Матеріали, що подаються до захисту БДР

До захисту подаються: пояснювальна записка БДР, графічні і ілюстративні матеріали, протокол попереднього захисту БДР на кафедрі, відгук наукового керівника, відгук рецензента, анотації до БДР українською та іноземною мовами, довідка про відповідність оформлення БДР діючим вимогам.

6 Перелік обов’язкового ілюстративного матеріалу

Під ілюстративним матеріалом маєм на увазі: презентацію, протокол попереднього захисту БДР на кафедрі, відгук наукового керівника, відгук рецензента, анотації до БДР українською та іноземною мовами, нормоконтроль про відповідність оформлення БДР діючим вимогам.

7 Порядок контролю та прийому

До приймання бакалаврської дипломної роботи надається:

— рубіжний контроль;

— попередній контроль;

— захист на ДЕК.

8 Вимоги до оформлювання та порядок виконання МКР

При оформлюванні МКР використовуються:

— ДСТУ 3008 : 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302 : 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— ГОСТ 2.104-2006 «Едина система конструкторської документації. Основні написи»;

— документи на які посилаються у вище вказаних.

Технічне завдання до виконання отримала _____ Колеснікова Н.С.

ДОДАТОК Б

Лістинг програми

```
import UIKit
enum MyTheme {
case    light    case
dark
}
var theme = MyTheme.dark
class ViewController: UIViewController {
        override func viewWillAppear(_ animated: Bool) {
            super.viewWillAppear(animated)
            self.noteTextView.text = ""
            if theme == .light { self.noteTextView.textColor
                = UIColor.black
            } else {
                self.noteTextView.textColor = UIColor.white
            }
        }
        override func viewDidAppear(_ animated: Bool)
            { super.viewDidAppear(animated)
            calenderView.changeTheme()
        }
        override func viewDidLoad()
            { super.viewDidLoad()
            noteTextView.layer.borderColor = UIColor(red: 0.9, green: 0.9, blue:
    0.9, alpha: 1.0).cgColor
            noteTextView.layer.borderWidth = 1.0
            noteTextView.layer.cornerRadius =
            5 self.title = "Календар"
            theme = .light
            Style.themeLight()
```

```
        self.navigationController?.navigationBar.isTranslucent=false
        self.view.backgroundColor=Style.bgColor
        view.addSubview(calenderView)
        calenderView.topAnchor.constraint(equalTo: view.topAnchor,
        constant:10
    isActive=true
        calenderView.rightAnchor.constraint(equalTo: view.rightAnchor,
    constant: .isActive=true
        calenderView.leftAnchor.constraint(equalTo: view.leftAnchor,
    constant:.isActive=true
        calenderView.heightAnchor.constraint(equalToConstant:
    365).isActive=true
        let rightBarBtn = UIBarButtonItem(title: "Темний", style: .plain,
    target: self, action: #selector(rightBarBtnAction))
        self.navigationItem.rightBarButtonItem = rightBarBtn
        NotificationCenter.default.addObserver(self,
selector:#selector(showProfile), NSNotification.Name("ShowProfile"),
    name: object: nil)NotificationCenter.default.addObserver(self,
                                                    selector:
    #selector(showArticles),                        name: object:
    NSNotification.Name("ShowArticle                nil)
    s"),
        NotificationCenter.default.addObserver(self,
                                                    selector:
    #selector(showDoctors),                         name: object:
    NSNotification.Name("ShowDoctor                 nil)
    s"),
        NotificationCenter.default.addObserver(self,
                                                    selector: #selector(share),
                                                    name:
    NSNotification.Name("Share"),
                                                    object: nil)
```

```
        NotificationCenter.default.addObserver(self,
                                               selector:
#selector(showHistory),                         name:
NSNotification.Name("History"),                 object: nil)
        NotificationCenter.default.addObserver(self,
                                               selector:
#selector(showPills),
NSNotification.Name("ShowPills"),
name:
object: nil)

NotificationCenter.default.addObserver(self,
                                               selector: #selector(rateUs),
                                               name:
NSNotification.Name("RateUs"),
                                               object: nil)
        NotificationCenter.default.addObserver(self,
                                               selector:
#selector(feedBack),                            name:
NSNotification.Name("FeedBack"                  object: nil)
),
        NotificationCenter.default.addObserver(self,
                                               selector:
```

```
 #selector(showAboutUs), NSNotification.Name("AboutUs"),
        }
        override func viewWillLayoutSubviews() { super.viewWillLayoutSubviews()
name: object: nil
 calenderView.myCollectionView.collectionViewLayout.invalidateLayout()
        }
        @objc func rightBarBtnAction(sender: UIBarButtonItem)
            { if theme == .dark {
                sender.title = "Темний"
                theme = .light
                Style.themeLight()
                self.noteTextView.textColor = UIColor.black
            } else {
                sender.title = "Світлий"
                theme = .dark
                Style.themeDark()
                self.noteTextView.textColor = UIColor.white
            }
            self.view.backgroundColor=Style.bgColo
            r calenderView.changeTheme()
            }
        let calenderView: CalenderView = {
            let v=CalenderView(theme: MyTheme.dark)
            v.translatesAutoresizingMaskIntoConstraints=false
            return v
        }()
        @objc func showProfile() {
            performSegue(withIdentifier: "ShowProfile", sender: nil)
        }
        @objc func showSettings() {
            performSegue(withIdentifier: "ShowSettings", sender: nil)
```

```
    }
    @objc func showArticles() {
        performSegue(withIdentifier: "ShowArticles", sender: nil)
    }
    @objc func showDoctors() {
        performSegue(withIdentifier: "ShowDoctors", sender: nil)
    }
    @objc func share() {
        let activityController = UIActivityViewController(activityItems: ["App
'Doctor Help' developed by Kateryna Levoshko"], applicationActivities: nil)
        self.present(activityController, animated: true)
    }
    @objc func showHistory() {
        performSegue(withIdentifier: "showHistory", sender: nil)
    }
    @objc func showPills() {
        performSegue(withIdentifier: "showPills", sender: nil)
    }
    @objc func rateUs() {
        let appstoreHooks = "itms-apps://itunes.apple.com/"
        let appstoreUrl = NSURL(string: appstoreHooks)
        if #available(iOS 10.0, *) { UIApplication.shared.open(appstoreUrl!
            as URL, options: [:],
completionHandler: nil)
} else {
            UIApplication.shared.openURL(appstoreUrl! as URL)
        }
    }
    @objc func feedBack() {
        let email = "klevoshko98@gmail.com"
        if let url = URL(string: "mailto:\(email)") { if
```

```
                #available(iOS 10.0, *) {
                    UIApplication.shared.open(url)
                } else {
                    UIApplication.shared.openURL(url)
                }
            }
        }
        @objc func showAboutUs() {
            performSegue(withIdentifier: "showAboutUs", sender: nil)
        }
        @IBAction func
            onMoreTapped() {
            print("TOGGLE SIDE
            MENU")
            NotificationCenter.default.post(name:
NSNotification.Name("ToggleSideMenu"), object: nil)
        }
        @IBOutlet weak var noteTextView: UITextView!
        override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
            if segue.identifier == "toNote" {
                if let destination = segue.destination as? NoteVC {
                    destination.cellFromCalendar = SavedNote.cell
                }
            }
            if segue.identifier == "toDoctorNote" {
                if let destination = segue.destination as? DoctorNoteVC {
                    destination.cellFromCalendar = SavedNote.cell
                }
            }
            if segue.identifier == "toPillNote" {
                if let destination = segue.destination as? PillNoteVC {
```

```
                    destination.cellFromCalendar = SavedNote.cell
                }
            }
        }
}
import UIKit import Firebase

class RegistrationVC: UIViewController, UITextFieldDelegate {
    let firebaseReference = Database.database().reference(withPath:
"users/")
    @IBOutlet weak var segmentedControl:
    UISegmentedControl! @IBOutlet weak var secondLabel:
    UILabel!
    @IBOutlet weak var thirdLabel: UILabel!
    @IBOutlet weak var nameTextField: UITextField!
    @IBOutlet weak var secondTextField:
    UITextField! @IBOutlet weak var thirdTextField:
    UITextField! @IBOutlet weak var emailTextField:
    UITextField! @IBOutlet weak var passTextField:
    UITextField!
    @IBOutlet weak var confirmTextField:
    UITextField! @IBOutlet weak var agreeButton:
    UIButton! @IBOutlet weak var signUpButton:
    UIButton!
    @IBAction func segmentedControlAction(_ sender:
        Any) { switch
        segmentedControl.selectedSegmentIndex { case 0:
            secondLabel.text = "Дата
            народження" thirdLabel.text =
            "Місто"
            brea
```

```
        k case 1:
            secondLabel.text = "Спеціалізація"
            thirdLabel.text = "Компанія"
            brea
        k default:
            break
        }
    }
    @IBAction func agreeButtonAction(_ sender: Any) {
    UIView.animate(withDuration: 0.2, delay: 0.0, options: .curveLinear,
animations: {
            self.agreeButton.transform = CGAffineTransform(scaleX: 0.1, y:
0.1)
        }) { (success) in
            self.agreeButton.isSelected = !self.agreeButton.isSelected
            UIView.animate(withDuration: 0.2, delay: 0.0, options:
.curveLinear, animations: {
                self.agreeButton.transform = .identity if
                self.agreeButton.isSelected &&
!self.nameTextField.text!.isEmpty &&
                    !self.secondTextField.text!.isEmpty &&
!self.thirdTextField.text!.isEmpty &&
                    !self.emailTextField.text!.isEmpty &&
!self.passTextField.text!.isEmpty &&
                    !self.confirmTextField.text!.isEmpty &&
                    self.passTextField.text == self.confirmTextField.text {
                    self.signUpButton.isEnabled = true
                } else {
                    self.signUpButton.isEnabled = false
                }
            }, completion: nil)
```

```
        }
    }
    @IBAction func signUpButtonAction(_ sender: Any) {
        Auth.auth().createUser(withEmail: emailTextField.text!, password:
passTextField.text!) { user, error in
            if error == nil {
                print("SUCCESS")
                let newUser = User(name: self.nameTextField.text!,
                                   db: self.secondTextField.text!,
                                   city: self.thirdTextField.text!,
                                   email: self.emailTextField.text!,
                                   weight: "-",
                                   height: "-",
                                   telephone: "-")
                let newUserReference =
self.firebaseReference.child("\(self.emailTextField.text!.replacingOccurrenc es(of:
".", with: ","))")
                newUserReference.setValue(newUser.toAnyObject())
                let userEmail = self.emailTextField.text!.replacingOccurrences(of:
                ".", with: ",")
                UserDefaults.standard.set(userEmail, forKey: "user")
                self.performSegue(withIdentifier: "fromSignupToCalendar",
sender: self)
            } else {
                var errorText = ""
                if error?.localizedDescription == "The email address is
badly formatted.
```

```
        errorText = "Неправильний формат вводу електронної пошти
 } else if error?.localizedDescription == "The email addressis already in use by
 another account." {
                        errorText = "Такий e-mail вже використовується
                        іншимкористувачем"
}
let alert = UIAlertController(title: "Помилка", message:errorText, preferredStyle:
.alert)
                    print(error?.localizedDescription)
                    let alertAction = UIAlertAction(title: "Ввести ще раз",
 style: .default, handler: nil)
                    alert.addAction(alertAction)
                    self.present(alert, animated: true, completion: nil)
                }
            }
        }
        override func viewWillAppear(_ animated: Bool) {
            super.viewWillAppear(animated)
            self.navigationController?.isNavigationBarHidden = false
        }
        override func viewDidLoad() {
            super.viewDidLoad()
            nameTextField.delegate = self
            secondTextField.delegate = self
            thirdTextField.delegate = self
            emailTextField.delegate = self
            passTextField.delegate = self
            confirmTextField.delegate = self
        self.navigationController?.isNavigationBarHi
            dden = false secondLabel.text = "Дата
```

```
        народження"
        thirdLabel.text = "Місто"
        passTextField.isSecureTextEntry = true
        confirmTextField.isSecureTextEntry = true
        [nameTextField, secondTextField, thirdTextField, emailTextField,
passTextField, confirmTextField].forEach({ $0.addTarget(self, action:
#selector(editingChanged), for: .editingChanged) })
    }
    @objc func editingChanged(_ textField: UITextField)
        { if textField.text?.characters.count == 1 {
            if textField.text?.characters.first == " " {
                textField.text = ""
                return
            }
        }
        guard
            let _ = nameTextField.text, !nameTextField.text!.isEmpty, let _
            = secondTextField.text, !secondTextField.text!.isEmpty, let _ =
            thirdTextField.text, !thirdTextField.text!.isEmpty, let _ =
            emailTextField.text, !emailTextField.text!.isEmpty, let _ =
            passTextField.text, !passTextField.text!.isEmpty,
            let _ = confirmTextField.text, !confirmTextField.text!.isEmpty,
            agreeButton.isSelected, confirmTextField.text ==
passTextField.text
            else {
                signUpButton.isEnabled =
                false return
        }
        signUpButton.isEnabled = true
    }
```

```
    func textFieldShouldReturn(_ textField: UITextField) -> Bool {
        textField.resignFirstResponder()
        return true
    }
    func textField(_ textField: UITextField, shouldChangeCharactersIn range:
    NSRange, replacementString string: String) -> Bool {

        if(string == "\n") {
            textField.resignFirstResponder()
            return false
        }
        if textField == secondTextField &&
segmentedControl.selectedSegmentIndex == 0 {
            let aSet = NSCharacterSet(charactersIn:"1234567890.").inverted
            let compSepByCharInSet = string.components(separatedBy:
            aSet) let numberFiltered =
            compSepByCharInSet.joined(separator: "") return string ==
            numberFiltered
        }
        return true
    }
}
```

ДОДАТОК В

Діаграма послідовності пошуку

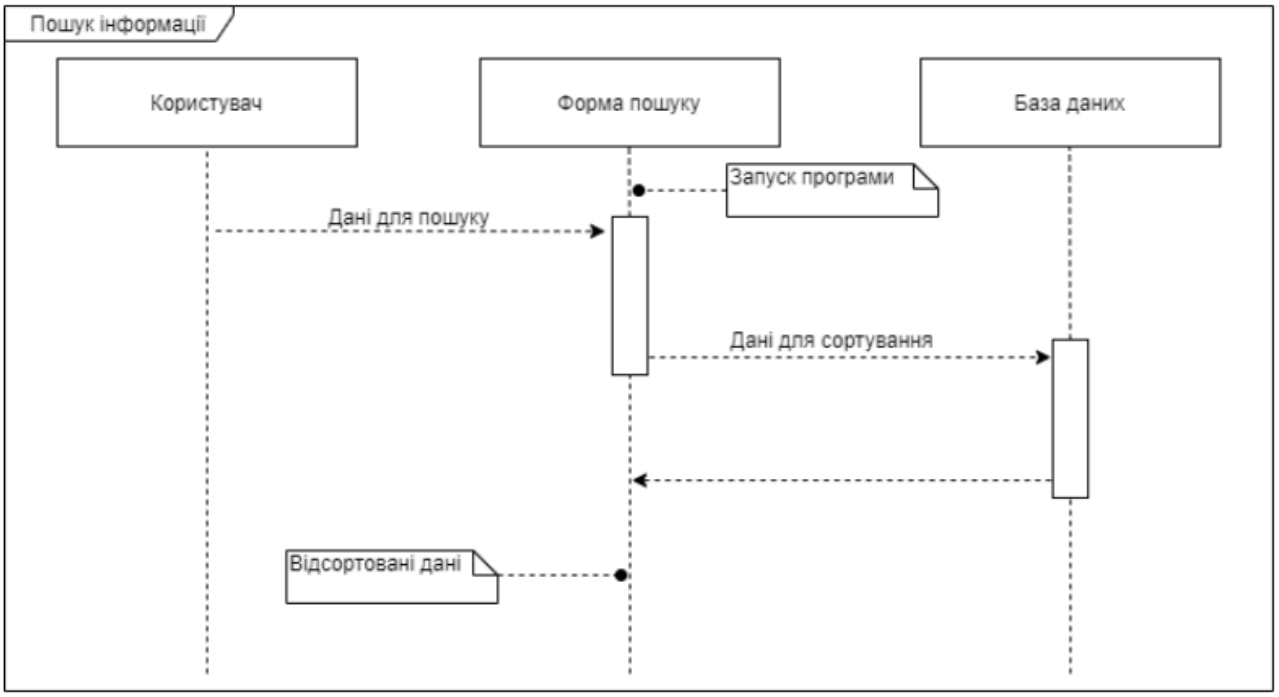


Рисунок В.1 — Діаграма послідовності пошуку

ДОДАТОК Г

Діаграма послідовності заявок

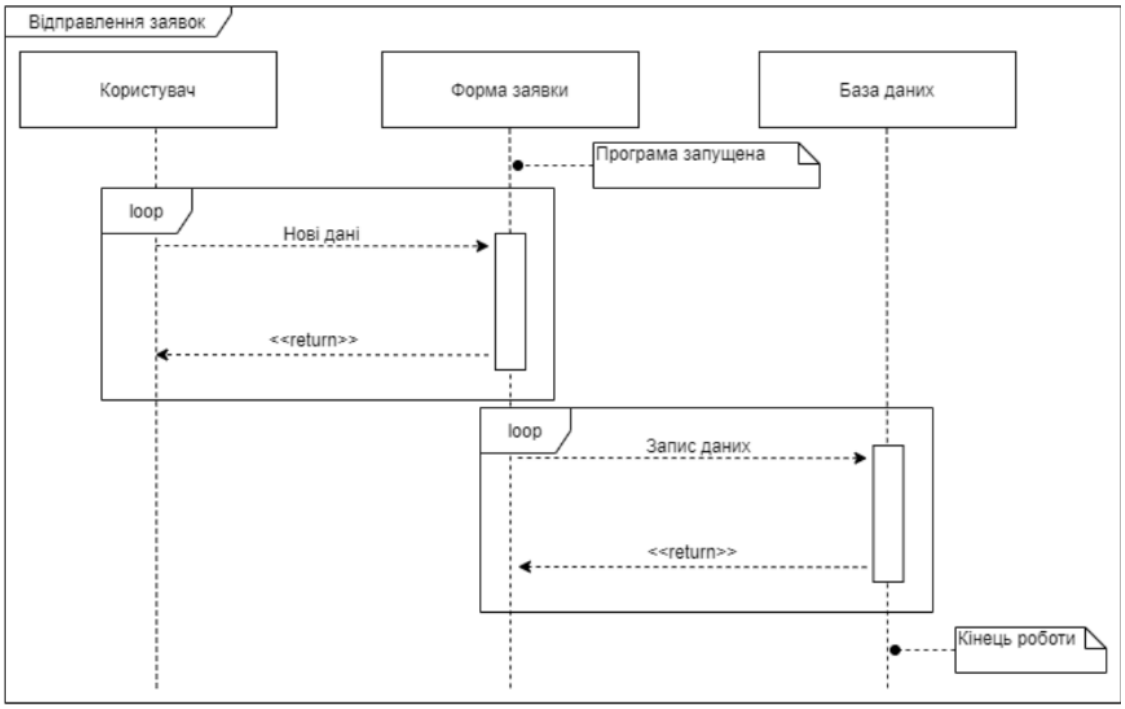


Рисунок Г.1 — Діаграма послідовності відправлення заявки

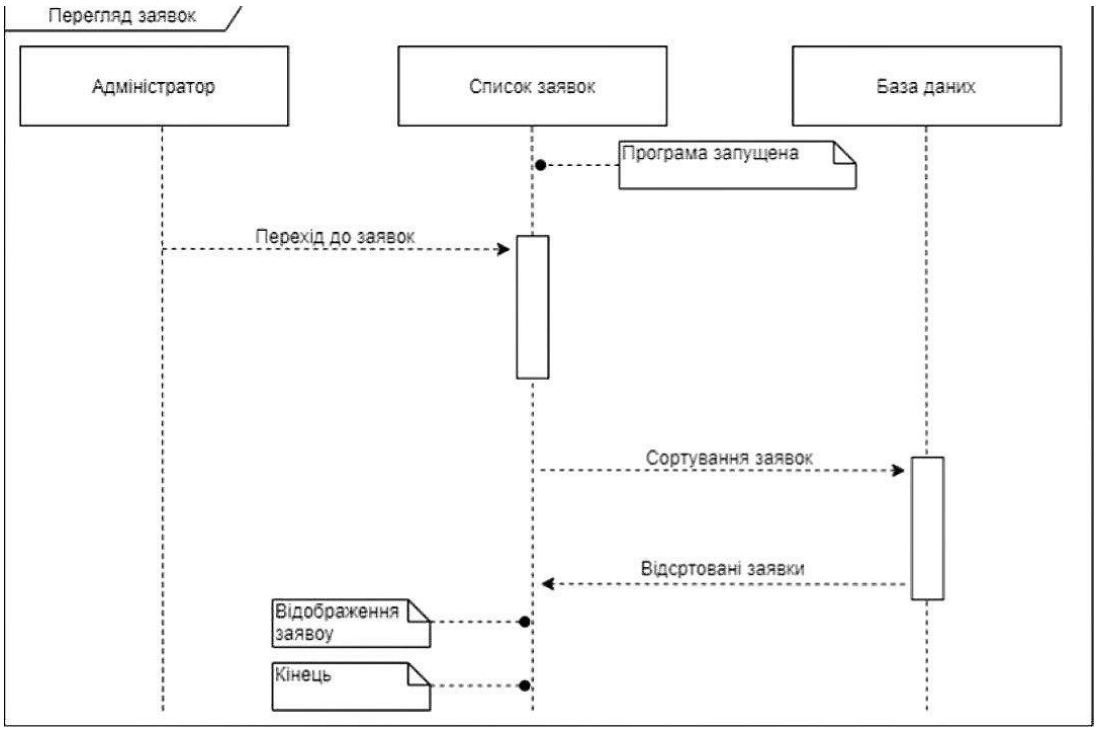


Рисунок Г.2 — Діаграма послідовності перегляду заявки

ДОДАТОК Д

Презентація

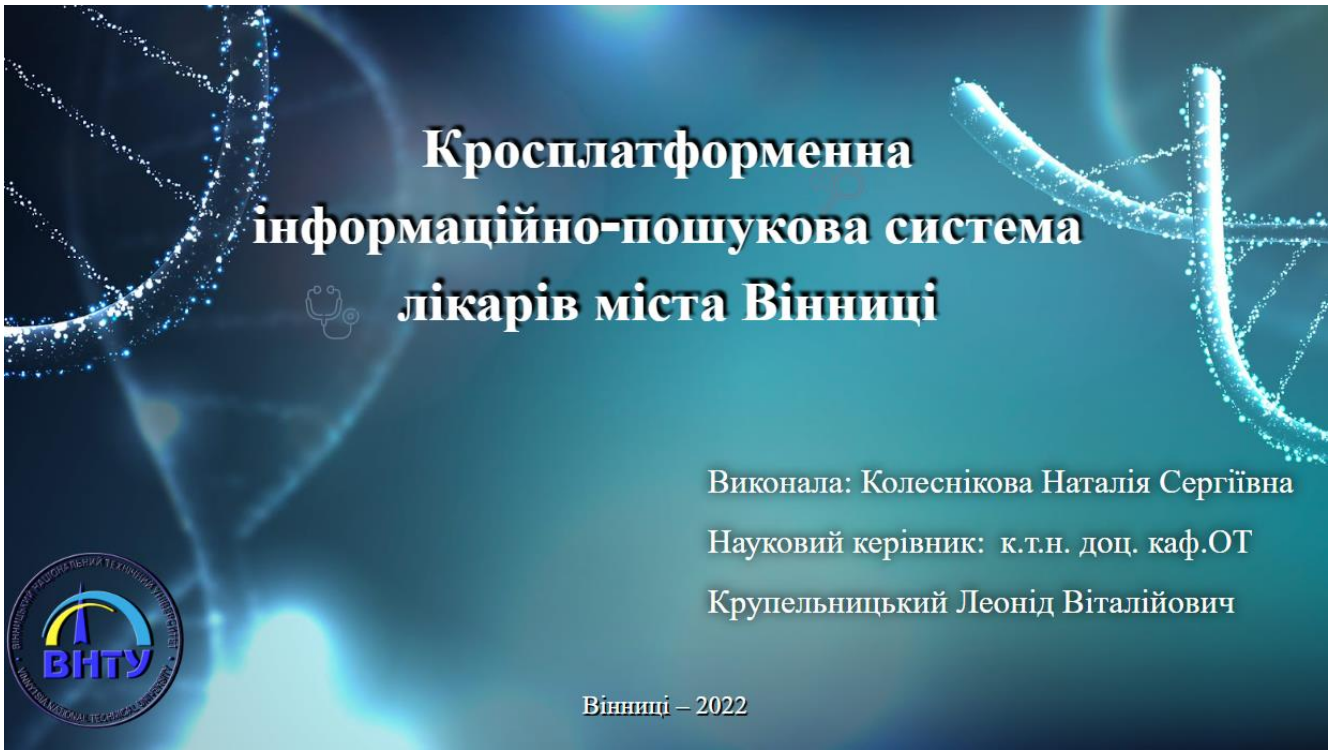


Рисунок Д.1 — Слайд 1

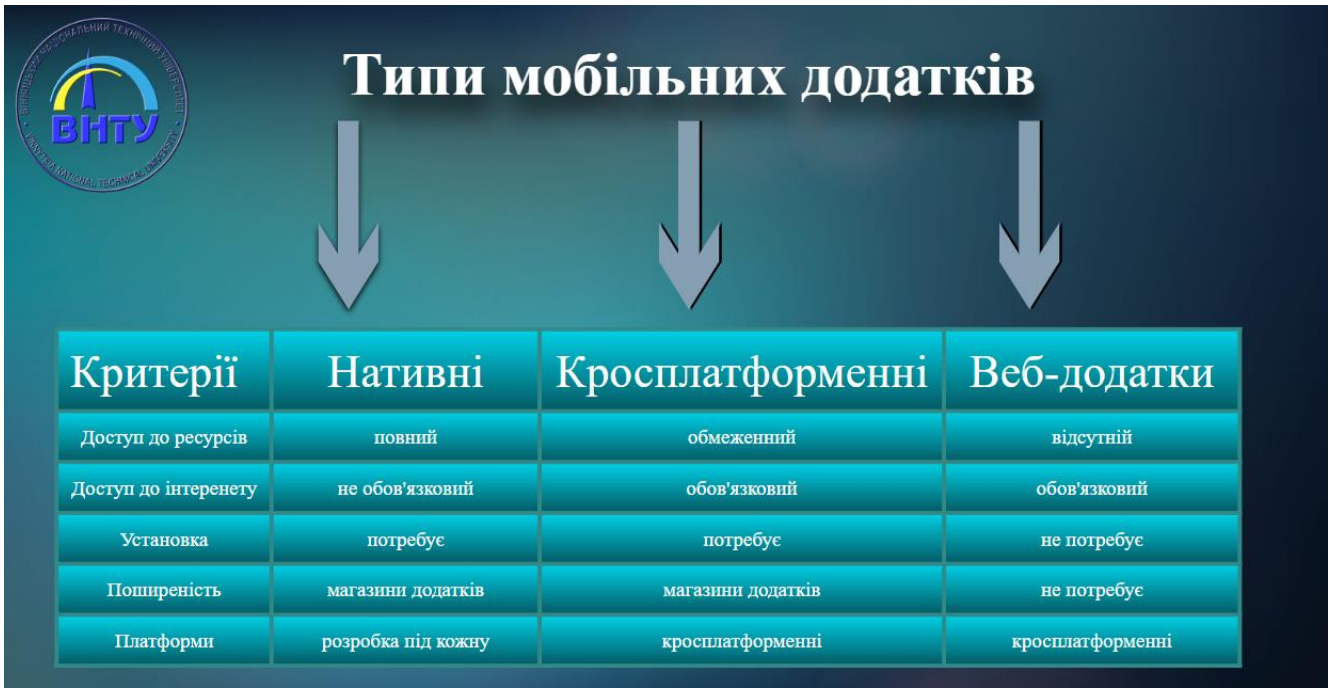


Рисунок Д.2 — Слайд 2

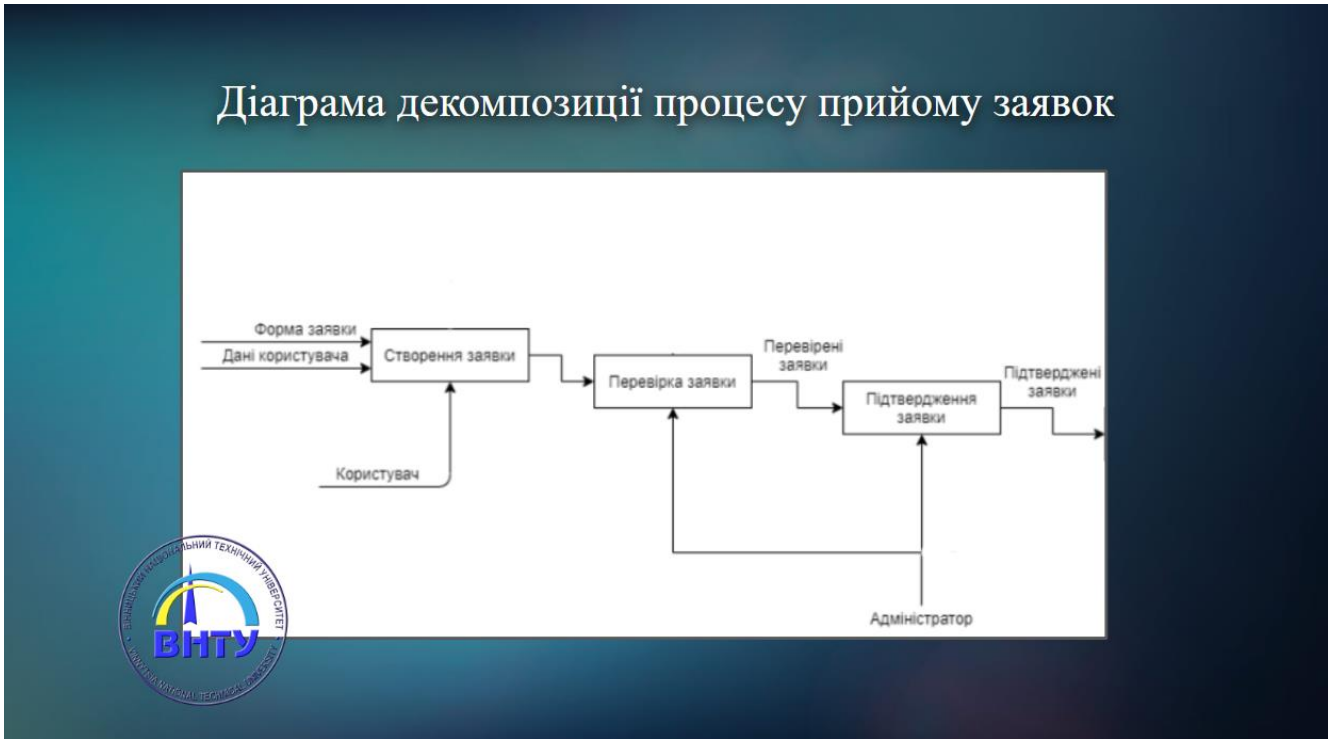


Рисунок Д.3 — Слайд 3

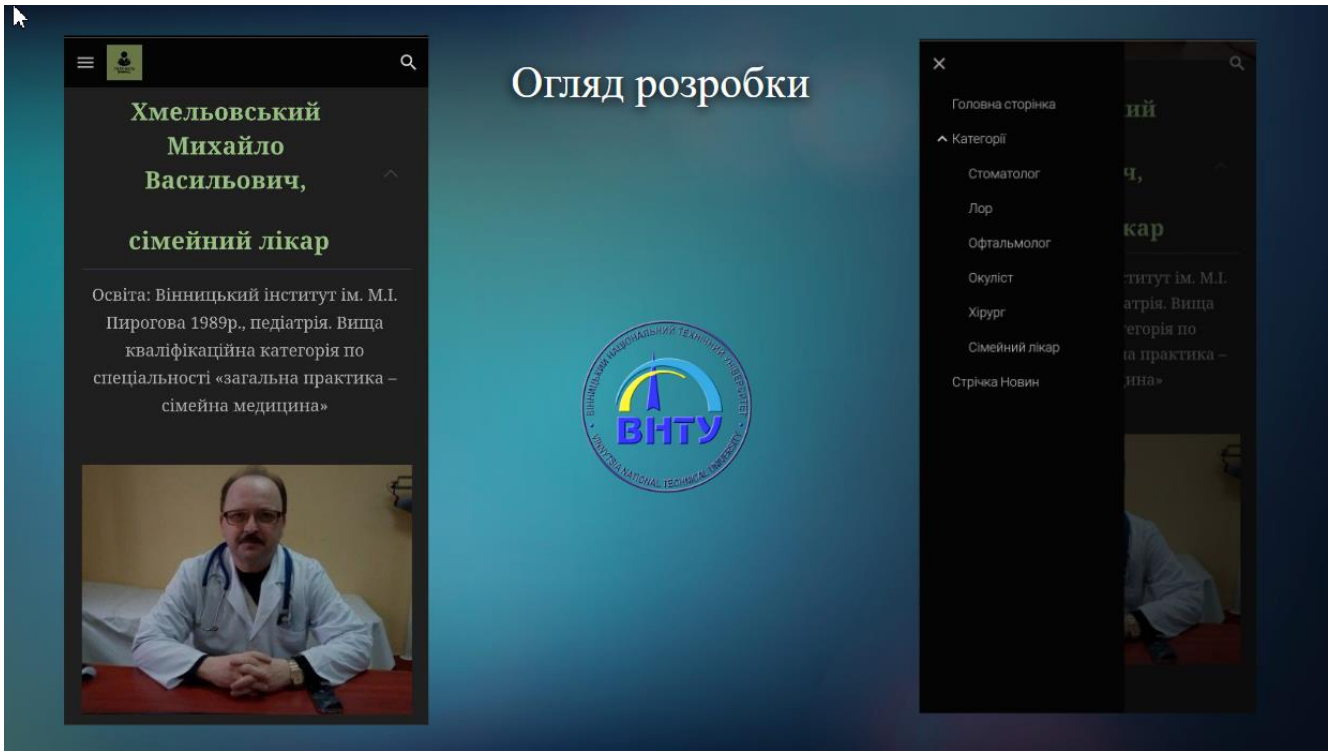


Рисунок Д.4 — Слайд 4

ДОДАТОК Ж

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ
РОБОТИНАНАЯВНІСТЬТЕКСТОВИХЗАПОЗ
ИЧЕНЬ

Назвароботи: Кросплатформенна інформаційно-пошукова система лікарів міста Вінниці.

Типроботи: бакалаврська дипломна робота

Підрозділ кафедра обчислювальної техніки

ПоказникизвітуподібностіUnicheck

Оригінальність 99% Схожість 1%

Аналіззвітуподібності(відмititипотрібне):

- ✓ Запозичення, виявлення у роботі, оформлені коректно і не містять ознак плагіату.
- □ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.
- □ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа,відповідальназаперевірку _____ Захарченко С.М.

Ознайомленізповнимзвітомподібності,якийбувзгенерованийсистемоюUnicheck щодороботи.

Авторроботи _____ Колеснікова Н.С.

Керівникроботи _____ Крупельницький Л.В.