НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

«До захисту допущено»

Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ
(підпис)
"___"_____2021 р.

Дипломний проєкт

на здобуття ступеня бакалавра

по спеціальності **123 «Комп'ютерна інженерія»**

на тему: Мобільний додаток для планування та контролю процесу лікування пацієнтів ветеринарної клініки

Виконала: студентка IV курсу, групи КВ-73

Романова Дар'я Володимирівна _____
(підпис)

Керівник ст. викл. Дробязко І.П. _____
(підпис)

Консультант з нормоконтролю доц.каф.СПСКС, к.т.н. Клятченко Я.М._____
(підпис)

Рецензент _____ _____
(підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2021 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп‘ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність 123 «Комп’ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ
(підпис)
«___»____2021 р.

ЗАВДАННЯ
на дипломний проєкт студента

Романової Дар’ї Володимирівни

1. Тема проєкту Мобільний додаток для планування та контролю процесу лікування пацієнтів ветеринарної клініки,

керівник проєкту Дробязко І.П., ст. викл. кафедри СПСКС,

затверджені наказом по університету від «25» травня 2021 р. №1331-С

2. Термін подання студентом проєкту від «27» травня 2021 р.

3. Вихідні дані до проєкту див. Технічне завдання

4. Зміст пояснювальної записки:

- Аналіз існуючих рішень та обґрунтування теми дипломного проєкту;
- Вибір технологічних та програмних засобів розроблення мобільного додатку;
- Опис розробленого мобільного додатку.

5. Перелік графічного матеріалу (із зазначенням обов’язкових креслеників, плакатів, презентацій тощо):

- Мобільний додаток для планування та контролю процесу лікування пацієнтів ветеринарної клініки. Схема структурна;

- Мобільний додаток для планування та контролю процесу лікування пацієнтів ветеринарної клініки. Діаграма класів;
- База даних. Схема структурна;
- Процедура обробки медичних карток. Схема алгоритму;
- Фрагменти програмного коду;
- Презентація дипломного проєкту.

6. Консультанти розділів проєкту*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Клятченко Я.М., доцент		

7. Дата видачі завдання 04.11.2020

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проекту	Примітка
1.	Видача завдання на дипломне проєктування	04.11.2020	
2.	Вивчення літератури за тематикою роботи	25.11.2020	
3.	Розроблення та узгодження технічного завдання	8.01.2020	
4.	Розроблення структури додатку та функціональних елементів	20.01.2021	
5.	Розроблення інтерфейсу додатку	10.02.2021	
6.	Програмна реалізація додатку	15.03.2021	
7.	Тестування додатку	04.04.2021	
8.	Підготовка матеріалів текстової частини проєкту	24.04.2021	
9.	Підготовка матеріалів графічної частини проєкту	17.05.2021	
10.	Оформлення технічної документації проєкту	25.05.2021	

Студент _____ Дар’я РОМАНОВА
(підпис) (ім’я, прізвище)

Керівник проєкту _____ Ірина ДРОБЯЗКО
(підпис) (ім’я, прізвище)

АНОТАЦІЯ

Кваліфікаційна робота включає пояснювальну записку (51с., 21 рис., 3 додатки).

Об’єкт розробки – мобільний Android-додаток для планування та контролю процесу лікування пацієнтів ветеринарної клініки.

Мета проєкту – підвищення ефективності роботи спеціалістів ветеринарної клініки.

Розроблений додаток дозволяє:

- здійснювати авторизацію спеціалістів ветеринарної клініки з розподілом прав доступу;
- створювати і підтримувати медичні картки пацієнтів;
- планувати процес лікування пацієнтів;
- визначати дозування лікарських препаратів;
- контролювати стан пацієнтів за показниками.

В процесі розробки використано мова програмування Kotlin, інструменти середовища розробки Android Studio. Для збереження даних використано Firebase Realtime Database.

В ході виконання дипломного проєкту:

- проведено аналіз існуючих програмних рішень для спеціалістів у галузі ветеринарії;
- проведено аналіз засобів створення мобільних Android-додатків;
- розроблено мобільний Android-додаток.

Використання додатку дозволить підвищити ефективність роботи ветеринарних спеціалістів завдяки покращенню взаємодії між лікарями та оперативному відслідковуванні стану пацієнтів.

Ключові слова: мобільний Android-додаток, Kotlin, Android Studio, Firebase Realtime Database.

ABSTRACT

Qualification work includes explanation note (51 pages, 21 drawings, 3 attachments).

The object of development is mobile Android application for planning and control of the treatment process of veterinary clinic patients.

The purpose of project is increasing the work efficiency of veterinary clinic's specialists.

Developed application enables:

- to carry out authorization of specialists of veterinary clinic with distribution of access rights;
- to create and support medical cards of patients;
- to plan patient's treatment;
- to determine the dosage of medical drugs;
- to control the condition of patients by indicators.

In the development process has been used programming language Kotlin, development environment Android Studio tools. For data saving has been used Firebase Realtime Database.

During the development of the diploma project:

- the analysis of existing software solutions for specialists in the field of veterinary medicine was carried out;
- the analysis of the ways of creating mobile Android applications was carried out;
- the mobile Android application was developed.

Using of application will allow to increase the work efficiency of veterinary specialists due to improving interaction between doctors and fast monitoring of patients' condition.

Keywords: mobile Android application, Kotlin, Android Studio, Firebase Realtime Database.

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Документація загальна			
			Новорозроблена			
	А4	ІАЛЦ.045490.002 ТЗ	Мобільний додаток для планування та контролю процесу лікування пацієнтів ветеринарної клініки Технічне завдання	4		
	А4	ІАЛЦ.045490.003 ТП	Мобільний додаток для планування та контролю процесу лікування пацієнтів ветеринарної клініки Відомість технічного проєкту	2		
	А4	ІАЛЦ.045490.004 ПЗ	Мобільний додаток для планування та контролю процесу лікування	51		

Змін	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045490.001 ОА		
Розробив		Романова Д.В.			Мобільний додаток для планування та контролю процесу лікування пацієнтів ветеринарної клініки **Опис альбому**	Літ.	Аркуш 1 / Аркушів 3
Перевірив		Дробязко І.П.					
Консульт.						КПІ ім. Ігоря Сікорського, ФПМ КВ-73	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Романкевич В.О.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			пацієнтів ветеринарної			
			клініки			
			Пояснювальна записка			
	А4	ІАЛЦ.045490.005 Д1	Мобільний додаток для	1		
			планування та контролю			
			процесу лікування			
			пацієнтів ветеринарної			
			клініки			
			Схема структурна			
	А4	ІАЛЦ.045490.006 Д2	Мобільний додаток для	1		
			планування та контролю			
			процесу лікування			
			пацієнтів ветеринарної			
			клініки			
			Взаємодія класів			
			Схема структурна			

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045490.007 Д3	Мобільний додаток для	1		
			планування та контролю			
			процесу лікування			
			ветеринарної клініки			
			База даних			
			Схема структурна			
	А4	ІАЛЦ.045490.008 Д4	Мобільний додаток для	1		
			планування та контролю			
			процесу лікування			
			ветеринарної клініки			
			Процедура обробки			
			медичних карток			
			Схема алгоритму			
		Диск CD-ROM	Мобільний додаток для	1		
			планування та контролю			
			процесу лікування			
			ветеринарної клініки			
			Текст пояснювальної			
			записки. Текст програми.			
			Графічний матеріал.			
			Презентація			

ЗМІСТ

					ІАЛЦ. 045490.002 ТЗ		
Змін	Арк.	№ докум.	Підпис	Дата			
Розробив		Романова Д.В.			Мобільний додаток для планування та контролю процесу лікування пацієнтів ветеринарної клініки **Технічне завдання**	Літ.	Аркуш 1 / Аркушів 4
Перевірив		Дробязко І.П.					
Н. контроль		Клятченко Я.М.				КПІ ім. Ігоря Сікорського, ФПМ КВ-73	
Затвердив		Романкевич В.О.					

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «Мобільний додаток для планування та контролю процесу лікування пацієнтів ветеринарної клініки».

Галузь застосування: інформаційні технології у галузі медицини.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання роботи першого (бакалаврського) рівня вищої освіти, затверджене кафедрою системного програмування і спеціалізованих комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проєкту є підвищення ефективності роботи спеціалістів ветеринарної клініки завдяки покращенню взаємодії між лікарями та швидкому відслідковуванні стану пацієнтів.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом інформації є науково-технічна література, технічна документація, публікації в періодичних виданнях та електронні статті у мережі Інтернет про існуючі аналоги.

5. ТЕХНІЧНІ ВИМОГИ

5.1 Вимоги до програмного продукту, що розробляється

- сумісність з операційною системою Android;
- можливість створення, редагування, видалення карток пацієнтів;
- впорядкування за категоріями інформації про існуючих пацієнтів;

- ведення історії хвороби, призначень лікаря та запис нотаток в процесі лікування;
- можливість авторизації окремо для різних категорій спеціалістів;
- планування та контроль проведення лікувальних процедур;
- інтуїтивно зрозумілий інтерфейс користувача.

5.2 Вимоги до апаратного забезпечення на стадії розробки

- наявність засобів доступу до мережі Wi-Fi (IEEE 802.11 b/g/n);
- наявність не менше 4 ГБ оперативної пам’яті.

5.3 Вимоги до програмного та апаратного забезпечення користувача

- операційна система Android;
- наявність доступу до мережі Wi-Fi (IEEE 802.11 b/g/n).

6. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів
1.	Видача завдання на дипломне проєктування	04.11.2020
2.	Вивчення літератури за тематикою роботи	25.11.2020
3.	Розроблення та узгодження технічного завдання	8.01.2020
4.	Розроблення структури додатку та функціональних елементів	20.01.2021
5.	Розроблення інтерфейсу додатку	10.02.2021
6.	Програмна реалізація додатку	15.03.2021
7.	Тестування додатку	04.04.2021
8.	Підготовка матеріалів текстової частини проєкту	24.04.2021
9.	Підготовка матеріалів графічної частини проєкту	17.05.2021
10.	Оформлення технічної документації проєкту	25.05.2021

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Документація загальна			
			Новорозроблена			
	А4	ІАЛЦ.045490.004 ПЗ	Мобільний додаток для	51		
			планування та контролю			
			процесу лікування			
			пацієнтів ветеринарної			
			клініки			
			Пояснювальна записка			
	А4	ІАЛЦ.045490.005 Д1	Мобільний додаток для	1		
			планування та контролю			
			процесу лікування			
			пацієнтів ветеринарної			
			клініки			
			Схема структурна			
	А4	ІАЛЦ.045490.006 Д2	Мобільний додаток для	1		
			планування та контролю			
			процесу лікування			
			пацієнтів ветеринарної			
			клініки			
			Діаграма класів			

Змін	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045490.003 ТП		
Розробив		Романова Д.В.			Мобільний додаток для планування та контролю процесу лікування пацієнтів ветеринарної клініки **Відомість технічного проекту**	Літ.	Аркуш 1 / Аркушів 2
Перевірив		Дробязко І.П.					
Консульт.						КПІ ім. Ігоря Сікорського, ФПМ КВ-73	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Романкевич В.О.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045490.007 ДЗ	Мобільний додаток для			
			планування та контролю			
			процесу лікування			
			пацієнтів ветеринарної			
			клініки			
			База даних.			
			Схема структурна			
	А4	ІАЛЦ.045490.008 Д4	Мобільний додаток для	1		
			планування та контролю			
			процесу лікування			
			пацієнтів ветеринарної			
			клініки			
			Процедура обробки			
			медичних карток.			
			Схема алгоритму			
		Диск CD-ROM	Мобільний додаток для	1		
			планування та контролю			
			процесу лікування			
			пацієнтів ветеринарної			
			клініки			
			Текст пояснювальної			
			записки. Текст програми.			
			Графічний матеріал.			
			Презентація			

Пояснювальна записка до дипломного проєкту

на тему: Мобільний додаток для планування та контролю процесу лікування пацієнтів ветеринарної клініки

Київ – 2021 року

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

Емулятор Android – програма емуляції функцій ОС Android в інших операційних середовищах

Кросплатформність – здатність програмного забезпечення працювати більш, ніж на одній апаратній платформі або операційній системі

СУБД – система управління базами даних

Android Debug Bridge (ADB) – засіб відлагодження та виявлення помилок для Android-додатків

API (Application Programming Interface) – програмний інтерфейс додатку

CRM (Customer Relationship Manager) – програмне забезпечення для автоматизації та управління взаємодією з клієнтами

URI (Uniform Resource Identifier) – уніфікований ідентифікатор ресурсу

ВСТУП

Із розвитком інформаційних технологій почалося впровадження комп'ютерних засобів у всі сфери життєдіяльності людини з метою підвищення її ефективності. Комп'ютерні технології відіграють провідну роль у різних галузях, також медицині, зокрема, ветеринарній справі.

Існує необхідність у впровадженні технічних і програмних рішень, що сприятимуть автоматизації робочого процесу ветеринарних спеціалістів та вирішенню різних проблем, що виникають під час роботи клініки. Кожний співробітник ветеринарної клініки має свої обов'язки. Виконання адміністративних функцій супроводжується використанням комп'ютерних систем, які працюють з документацією, бухгалтерією, допомагають у керуванні кадрами, взаємодією з клієнтами та систематизацією інформації. За допомогою сучасних програм здійснюється та контролюється якісне надання послуг. Для ефективної роботи ветеринарних лікарів існує спеціалізоване забезпечення для медичної діагностики, терапії та проведення операційних втручань. Для контролю стану пацієнтів стаціонару необхідні окремі комп'ютерні засоби, що зберігають інформацію про стан здоров'я на кожному етапі лікування. Сучасна ветеринарна медицина має потреби у автоматизації процесу обробки інформації: електронному опрацюванні медичних карток пацієнтів, централізованому збереженні та використанні довідникової інформації, швидкому доступі до інструментів, що спряють оперативному прийняттю рішень спеціалістами в процесі роботи.

На програмні засоби, що використовуються у галузі ветеринарії, накладаються такі вимоги, як мобільність, доступність та зручність інтерфейсу, швидкість роботи, наявність функціональних можливостей відповідно до особливості роботи різних спеціалістів.

Метою дипломного проекту є створення мобільного Android-додатку для спеціалістів ветеринарної клініки, що, завдяки покращенню взаємодії між

лікарями та швидкому відслідковуванні стану пацієнтів, дозволить підвищити ефективність роботи спеціалістів.

1 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ОБҐРУНТУВАННЯ ТЕМИ ДИПЛОМНОГО ПРОЄКТУ

1.1 Особливості програмних рішень у сфері ветеринарної медицини

Існує багато програм та мобільних додатків для лікарів та спеціалістів у сфері ветеринарної медицини, що реалізують різні функції та полегшують роботу фахівців. У клініках використовують програми, що дозволяють автоматизувати документообіг, включаючи ведення електронних медичних карток пацієнтів, облік та планування прийомів у лікаря, проведення таких операцій із документами, як складання документів, їх зберігання та друк. Також більшість серверів підтримують функції взаємодії з клієнтами, а саме: розсилка SMS та e-mail, нагадування тощо. Практикуючі спеціалісти отримують доступ до атласів, енциклопедій та ветеринарної системи КАД (комп'ютерно-асистованої діагностики), яка відноситься до засобів швидкої диференціальної діагностики.

Найбільшої популярності набули CRM-системи (Customer Relationship Management) – програмне забезпечення для автоматизації та управління взаємодією з клієнтами. Основна ідея CRM – об'єднання різних бізнес-інструментів у цілісну систему, яка спрямована на автоматизацію усіх процесів надання послуг, контроль шляхів комунікації та збір інформації про клієнтів. До переваг використання таких систем можна віднести наступні:

1) єдина база клієнтів, де зберігаються всі зібрані дані;
2) облік послуг, що дозволяє формувати статистику, виконувати аналіз для покращення розвитку підприємства;
3) інтеграція з бухгалтерією, що дозволяє обмінюватися даними в автоматичному режимі;
4) контроль роботи працівників;
5) забезпечення нагадувань про заплановані події.

Вищезазначені функції можуть варіюватися та комбінуватися у різних CRM-системах, але загальний концепт залишається незмінним.

До недоліків можна віднести деякі складнощі, що виникають з освоєнням нової системи та періодом адаптації до збільшених функціональних можливостей. Тому зрозумілість інтерфейсу та доступність використання основних інструментів є важливими аспектами у сучасних медичних програмах.

З іншого боку, автоматизація обробки документів має свої проблеми, оскільки внесення помилкових даних у документи призводить до невідповідного лікування та непередбачуваних наслідків.

Ветеринарні клініки працюють з програмами як універсальними, що містять стандартний функціонал без спеціального призначення та прив’язки до медичної сфери діяльності, так і спеціальними для галузі ветеринарії.

Аналіз таких відомих програм для ветеринарних клінік, як VetDesk, Vetmanager, VetAIS, VetCliniX, Ветеринарний офіс, свідчить про те, що дані можуть зберігатися або у хмарних сервісах, або на власних серверах підприємства, що передбачає покупку ліцензії на використання програми та встановлення на комп’ютери. Найбільш поширеним є користування онлайн сервісами через мережу Інтернет, що уможливлює доступ з будь-якого комп’ютера. В багатьох випадках можливо встановити додаток на смартфон, персональний комп’ютер та інші гаджети або заходити через браузер. Це забезпечує достатню мобільність для вирішення поставлених задач.

Аналізуючи існуючі програмні рішення для швидкого доступу до впорядкованої інформації у вигляді довідників, прослідковується тенденція актуальних для ветеринарних спеціалістів питань. Наприклад, точне дозування препаратів для відповідної маси тіла тварини та інших вимірів або розрахунок інфузії для крапельниць.

Зважаючи на це, виникає потреба у додатках, що забезпечують швидкий доступ до довідкової інформації з догляду за тваринами, раціоном харчування, щодо методів лікування хвороб та паразитів, лікувальних препаратів тощо, та надають додаткові послуги, наприклад, спеціалізовані калькулятори.

1.2 Аналіз існуючих мобільних додатків для ветеринарних спеціалістів

1.2.1 Vetmanager Mobile

Одним із найбільш популярних програмних рішень є Vetmanager – сучасна хмарна система для керування такими структурами у галузі ветеринарії, як ветеринарні клініки, аптеки та зоомагазини, а також для оптимізації ветеринарної практики та впровадження CRM-технології [1]. Vetmanager містить широкий спектр інструментів для ефективної роботи різних працівників – ветеринарних лікарів та адміністраторів в умовах праці великих мереж і малих клінік. Прикладом може слугувати мобільний додаток для лікарів Vetmanager Mobile, що забезпечує необхідною для роботи інформацією та швидкий доступ до його сервісів з мобільного пристрою на базі Android.

Електронні медичні картки у повній мірі замінюють використання паперових носіїв інформації. З огляду на те, що зберігання карток пацієнтів у електронному вигляді уможливлює доступ до даних з різних носіїв, швидкий пошук, додавання результатів аналізів, знімків, фото та відеозаписів, такий вид збереження є найбільш зручним та ефективним. Підтримуються також заздалегідь створені шаблони для огляду та лікування пацієнтів.

Vetmanager Mobile підтримує такі основні сторінки, як:

1) домашня;
2) візити;
3) дзвінки;
4) стаціонар;
5) налаштування.

Інтерфейс мобільного додатку Vetmanager Mobile представлено на рис.1.

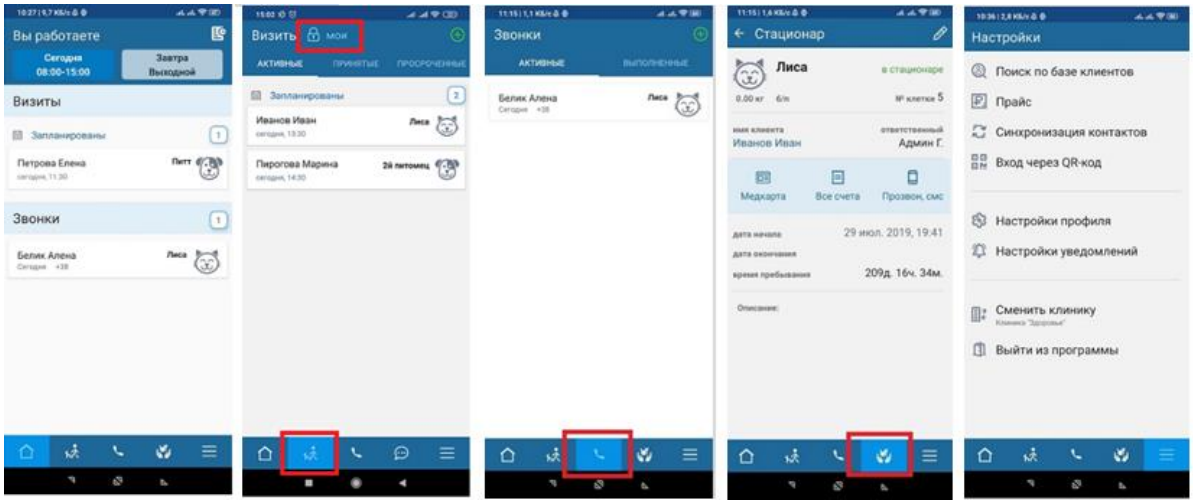

Рисунок 1 – Інтерфейс Vetmanager Mobile

Мобільний додаток Vetmanager Mobile забезпечує наступні функціональні можливості:

1) авторизація за доменом клініки, персональним логіном та паролем;
2) перегляд графіку роботи працівників з нагадуванням;
3) отримання інформації про заплановані візити та дзвінки від клієнтів;
4) можливість створення та доступу до медичних карток пацієнтів, рахунків;
5) робота з клієнтами через відслідковування початку та закінчення прийому лікаря;
6) планування та збереження історії візитів, дзвінків;
7) отримання інформації про пацієнтів, що перебувають у стаціонарі;
8) пошук у базі клієнтів та цін на послуги;
9) налаштування сторінки користувача та повідомлень;
10) вхід у програму через браузер за допомогою QR-коду.

До недоліків додатку можна віднести особливості використання хмарних програм – залежність від доступу до мережі Інтернет, заплановані технічні роботи на сервері тощо. Також після авторизації лікар має доступ лише до тих пацієнтів, що закріплені за ним.

1.2.2 Veterinary Emergency Medicine Small Animal

Зважаючи на необхідність швидкого доступу ветеринарних лікарів до професійної інформації при наданні невідкладної допомоги, серед фахівців поширене використання мобільних додатків, які виконують функцію довідників. Такі програми містять дані про лікування окремих видів тварин та спрямовані на вирішення проблем відповідних фахівців. Оскільки лікар-хірург, медична сестра, спеціаліст з догляду та лікування великого рогатого скоту або екзотичних тварин мають різні потреби, існує достатньо великий вибір необхідних додатків: Vet Nurse Quick Reference, Veterinary Handbook, Veterinary Care of Exotic Pets тощо.

Мобільний додаток Veterinary Emergency Medicine Small Animal розроблено спеціалістом ветеринарії дрібних тварин та невідкладної допомоги Шайліном Джасані [2]. Додаток містить ґрунтовну інформацію, що є здобутком досвіду викладацької та практичної діяльності фахівців, і зібрана для створення ресурсу допомоги у клінічній практиці.

Додаток надає наступні можливості:

1) калькулятори дозування препаратів;
2) калькулятори рідинної терапії;
3) формули найбільш поширених медичних препаратів з показаннями, дозуванням та інструкцією;
4) коротка інформація про ускладнення, що можуть зустрічатися в процесі лікування;
5) дані для перевірки фізіологічних показників пацієнта;
6) бібліотека знімків, аналізів, кардіограм та ін.

Мобільний додаток Veterinary Emergency Medicine Small Animal особливий тим, що виконує функцію довідника та може слугувати підручником з нотатками для молодих спеціалістів, студентів та практикуючих лікарів. Калькулятори лікарських препаратів є дуже зручними у повсякденному використанні та застосовуються у багатьох клініках, полегшуючи розрахунки.

Недоліком таких додатків є їх вузька спрямованість на роботу з окремими видами тварин, з огляду на великий обсяг необхідної для цього інформації. Також наявні дані повинні оновлюватися по мірі розвитку знань у сфері ветеринарної медицини та нової актуальної проблематики.

Veterinary Emergency Medicine Small Animal має достатньо простий інтерфейс, який представлено на рис. 2.

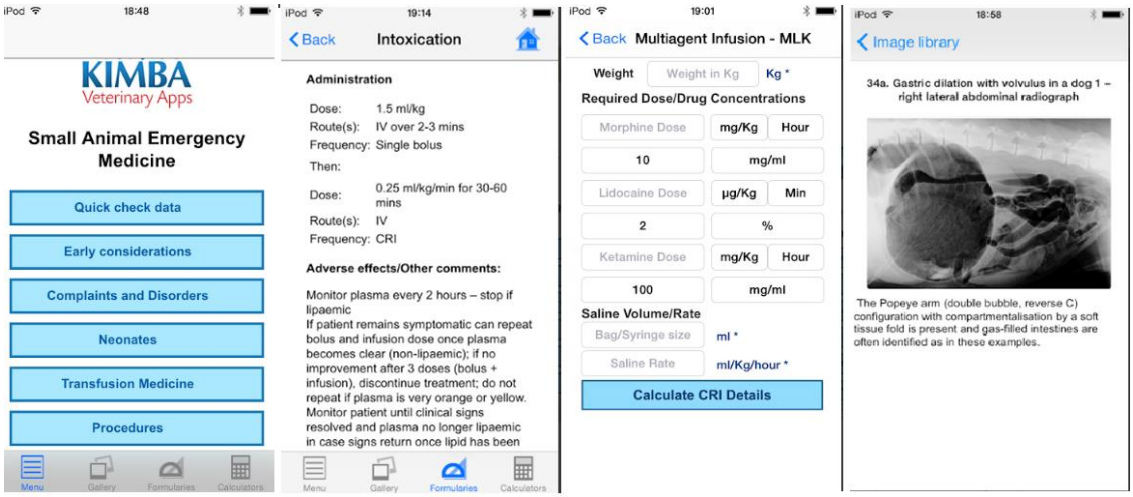

Рисунок 2 – Інтерфейс Veterinary Emergency Medicine Small Animal

1.2.3 Vetcalculators

У випадках, коли ветеринарному лікарю необхідно виконати точні розрахунки максимально швидко, через специфіку роботи у невідкладних умовах, застосовуються мобільні додатки – калькулятори. Додаток Vetcalculators має сукупність інструментів для підрахунку як дозування препаратів, так і показників фізіологічного стану пацієнтів. Він є універсальним для використання у різних напрямах лікування та ветеринарів усіх спеціалізацій. До переваг можна віднести доступ до всіх функцій без необхідності підключення до мережі Інтернет [3].

Додаток Vetcalculators надає наступні можливості:

1) 13 калькуляторів для розрахунку препаратів, вимірювань показників крові, глюкози, токсинів, доповнення власними ліками;

2) автоматичне корегування вихідних даних при зміні показників стану пацієнта;

3) вибір потрібних для роботи калькуляторів;

4) збереження даних розрахунку, робота в офлайн режимі;

5) сканування документів та друк лікарських доз.

Даний мобільний додаток розрахований на англомовних користувачів, тому можуть виникнути відповідні проблеми з використанням його функцій за відсутності певного рівня знання англійської мови.

Інтерфейс додатку Vetcalculators представлено на рис. 3.

Рисунок 3 – Інтерфейс Vetcalculators

1.2.4 ViralVet – Veterinary Cases

Головне завдання мобільного додатку ViralVet – створення середовища для комунікації спеціалістів у сфері ветеринарії. Програма містить функціонал для ведення дискусій із практикуючими фахівцями та розміщення контенту, що містить навчальний характер та представляє науковий інтерес для лікарів. Тому особливістю ViralVet є забезпечення ветеринарних лікарів вузько спрямованою соціальною мережею задля співпраці та професійного росту [4].

ViralVet має наступні можливості:

1) автентифікація користувачів із визначенням статусу спеціаліста у ветеринарній сфері – лікар, технолог, студент;

2) перегляд та пошук зображень, відео, записів, що відфільтровані за видами та спеціалізацією;

3) функція розміщення власних спостережень, питань до фахівців, випадків з практики;

4) запрошення знайомих та приєднання до групи друзів.

Мобільний додаток ViralVet забезпечує взаємодію між спеціалістами різного професійного рівня, але ніяким чином не впливає на робочий процес всередині клініки. На користувачів додатку не накладається відповідальність за встановлення невірного діагнозу та поради у випадку недостатньої кваліфікації.

Інтерфейс ViralVet представлено на рис. 4.

Рисунок 4 – Інтерфейс ViralVet

1.3 Обґрунтування теми дипломного проєкту

Аналіз існуючих систем та мобільних додатків показує, що існуючі програмні рішення систем у сфері ветеринарної медицини, які забезпечують взаємодію лікарів, є багатофункціональними, проте містять надмірну кількість інструментів для вирішення повсякденних задач ветеринарних лікарів, і в той самий час є більш орієнтованими на потреби адміністративного відділу клініки. Також вони вимагають додаткових витрат з боку клініки через ліцензованість та особливості використання хмарних сервісів. Довідники ж та додатки-калькулятори мають вузьку направленість, є негнучкими та надають невеликі функціональні можливості.

Вищезазначене доводить актуальність створення зручного у використанні мобільного додатку для працівників ветеринарної клініки з функціями планування процесу лікування та контролю за станом пацієнтів.

Метою розробки мобільного додатку є підвищення ефективності роботи фахівців клініки шляхом покращення взаємодії між лікарями та швидкому відслідковуванні стану пацієнтів.

Додаток забезпечуватиме наступні функціональні можливості:

1) аутентифікація ветеринарних лікарів та асистентів;
2) створення і підтримка медичних карток пацієнтів;
3) планування проведення операцій та процедур;
4) додавання записів лікаря для діагностування та контролю за процесом лікування;
5) розрахунок дозувань лікарських препаратів;
6) внесення результатів вимірювань – показників фізичного стану пацієнтів та результатів аналізів.
7) інтуїтивно зрозумілий і зручний у використанні інтерфейс користувача.

2 ВИБІР ТЕХНОЛОГІЧНИХ ТА ПРОГРАМНИХ ЗАСОБІВ РОЗРОБЛЕННЯ МОБІЛЬНОГО ДОДАТКУ

2.1 Операційна система

Android – операційна система (ОС) для мобільних пристроїв, яка базується на модифікованому ядрі Linux і реалізації віртуальної Java машини від Google. Розроблена Android Inc. та придбана у 2005 році компанією Google [5]. Використання розроблених Google-бібліотек забезпечує керування пристроєм через Java-додаток. ОС Android є системою відкритого типу – такою, що дозволяє редагування, має відкритий вихідний код. Це зумовлює швидкий розвиток та гнучкість налаштування під необхідний пристрій, контроль за помилками з боку користувачів [6].

ОС Android має ряд версій, кожна з яких має свої особливості у порівнянні з попередніми, що представлено на рис. 5 [7].

Рисунок 5 – Позначення версій ОС Android

Для створення додатків для пристроїв на базі Android використовується набір інструментів у комплекті Android SDK (Software Development Kit), таких, як емулятор, Android Debug Bridge (ADB), набір бібліотеки додаткового коду для Java-програм. SDK засіб розробки мобільних додатків забезпечує тестування, налагодження програм у режимі сумісності з різними версіями платформи

Android та перегляд у режимі реального часу на віртуальному пристрої. Також Android SDK є крос-платформним та може бути використаний на комп'ютерах із різною ОС. На поточний час Android SDK включено у середовище розробки Android Studio, яке широко застосовується для роботи з платформою Android. Для написання програм та розробки мобільних додатків застосовується JDK (Java Development Kit), що містить компілятор Java, різноманітні стандартні бібліотеки класів, документацію, виконавчу систему Java [8].

2.1.1 Архітектура ОС Android

Архітектура ОС Android складається з таких рівнів:

1) додатків (Applications) – сукупність основних додатків, таких, як браузер, мапи, календар, SMS і e-mail-клієнти тощо;

2) фреймворку додатків (Application Framework) – структура, що дозволяє використовувати можливості одного додатка іншими, що мають доступ до нього – принцип багаторазового використання компонентів;

3) бібліотек (Libraries) – C/C++ бібліотеки, що використовуються компонентами ОС, та функції яких доступні через рівень фреймворку додатків [9]:

- System C Library – реалізація стандартної C бібліотеки;
- Media Libraries – бібліотеки для виконання та запису більшості відео- та аудіо-форматів;
- Surface Manager – менеджер, що керує доступом до відображення 2D і 3D графічних прошарків системи;
- LibWebCore – бібліотека, яка дозволяє користуватися можливостями вбудованого браузера Android;
- SGL – інструмент, який дозволяє працювати з 2D-графікою;
- 3D libraries – бібліотеки, які дозволяють працювати з 3D-графікою;
- FreeType – бібліотека для роботи зі шрифтами;

- SQLite – потужний двигун для роботи із реляційними базами даних.

4) середовища виконання (Android Runtime) – застосування платформою регістр-орієнтованої віртуальної машини Dalvik, що використовує формат .dex, у який компілюється байт-код Java за допомогою SDK, і кожний екземпляр якої супроводжує запуск додатку з власним процесом [10];

5) ядра Linux (Linux Kernel) – доступ до системних служб ядра ОС Linux – управління процесами та пам'яттю, забезпечення безпеки, роботи з драйверами та мережею.

Архітектуру ОС Android представлено на рис. 6.

Рисунок 6 – Архітектура ОС Android

Додатки ОС Android мають свої особливості:

1) додатки обмінюються даними (Content providers);
2) додатки мають доступ до ресурсів – медіа-файлів (Resource Manager);
3) виконується управління активностями додатків (Activity Manager);
4) виконується контроль за станом додатку (Notification Manager).

За допомогою SDK із скомпільованих даних та ресурсів створюється архів формату .apk для встановлення мобільного додатку на пристрій [11].

2.1.2 Компоненти Android-додатку

Android-додаток будується зі складових, які називаються компонентами, кожний з яких має своє призначення, життєвий цикл. Виділяють наступні види:

1) активності (Activities);
2) служби (Services);
3) широкомовні приймачі (Broadcast receives);
4) контент-провайдери (Content providers).

Активності виконують функцію взаємодії з користувачем через інтерфейс додатку. Android-додаток може мати декілька окремих активностей, які мають різне призначення та дозволяють отримувати доступ до таких системних можливостей:

1) відслідковування дій користувача у поточній активності;
2) збереження інформації про попередні активності та забезпечення повернення до них;
3) збереження даних поточної активності під час роботи з новою активністю;
4) забезпечення коректної роботи з потоками.

Служби забезпечують фонову роботу довгострокових операцій та віддалених процесів. Служби розрізняють як ті, що є явними та підтримують стабільну роботу, яка необхідна користувачу, і фонові, що не мають прямого контакту з користувачем і можуть бути перерваними системою. Інтерфейс у випадку використання служб не передбачений. Наприклад, відтворення музики у фоновому вигляді.

Широкомовні приймачі дозволяють отримувати зовнішні події та відповіді на ці події. Існує можливість прослуховувати широкомовні повідомлення інших додатків і в результаті виконувати зміни у функціональності власного додатку.

Наприклад, отримання повідомлень незалежно від стану роботи – у фоновому режимі або його відсутності. Існують види широкомовних повідомлень:

1) явні, які призначаються конкретному ряду додатків, та знаходять отримувача за target-атрибутом;
2) неявні, які передаються на всі доступні додатки, які готові прийняти трансляцію, та не мають атрибута цілі.

Для того, щоб користуватися необхідними для додатку даними, що містяться у ОС, існують контент-провайдери. Наприклад, медіа-файли, налаштування, словники, календарі, контакти тощо. Контент-провайдери забезпечують постачання даних у інші додатки системи, які потребують їх. Доступ можна отримати за запитами URI для подальшого виконання операцій. Окрім використання вбудованих контент-провайдерів, існує можливість створення власних.

Для того, щоб пов’язати різні компоненти у процесі їх виконання, використовують наміри (Intents). Незалежно від того, чи належить компонент даному додатку, наміри роблять запити для активації цього компоненту або визначеного типу, що розподіляє наміри на явні та неявні. У випадку активностей і служб, визначаються задачі для виконання з доступом до необхідних даних через запити URI. Таким чином активності можуть повертати результат виконання через намір. Широкомовні приймачі використовують наміри для визначення повідомлення, що передається. Наміри не потрібні для активації контент-провайдерів, оскільки ті мають власні методи активації через виконання запиту [12].

Для того, щоб попередити систему про існування будь-якого компоненту, необхідно його оголосити у файлі AndroidManifest.xml. Окрім цього, у файлі маніфесту визначаються такі додаткові параметри:

1) системні функції, що використовуються додатком;
2) усі необхідні дозволи користувача для використання служб;
3) мінімальний рівень API додатку;
4) бібліотеки API.

Додаток потребує такі ресурси , як програмний код, медіа-ресурси. Для того, щоб отримати доступ до зображень, стилів, рядків, кольорів тощо, які визначаються за допомогою XML-файлів, кожний з них має власний ідентифікатор. Також для різних конфігурацій додатка існує можливість використання альтернативних ресурсів, впровадження кваліфікаторів для визначення конкретної кваліфікації.

2.1.2 Життєвий цикл Android Activity

Компонент Activity має свій життєвий цикл, який описує стани, що набувають змін впродовж роботи додатку. Інформація про всі зміни стану активності забезпечується зворотними викликами, а саме, створенням, призупиненням, продовженням, завершенням процесу діяльності.

Зворотні виклики дозволяють вживати заходів за певних змін у стані активності. Таким чином, вирішуються проблеми використання ресурсів користувачем, передача даних для збереження поточного прогресу, обробка ситуацій, де виникають збої. Для того, щоб перехід між етапами життєвого циклу здійснювався коректно, при кожному переході викликаються зворотні виклики.

Для запуску активності вперше застосовується виклик onCreate(). З початком виконання активності у початковому стані викликається onStart(), який змінюється на onPause(), onResume() та onStop() у випадку призупинення, відновлення та остаточної зупинки активності відповідно. Зворотній виклик onDestroy() викликається наприкінці життєвого циклу активності [13].

Спрощене зображення життєвого циклу Android активності представлено на рис. 7.

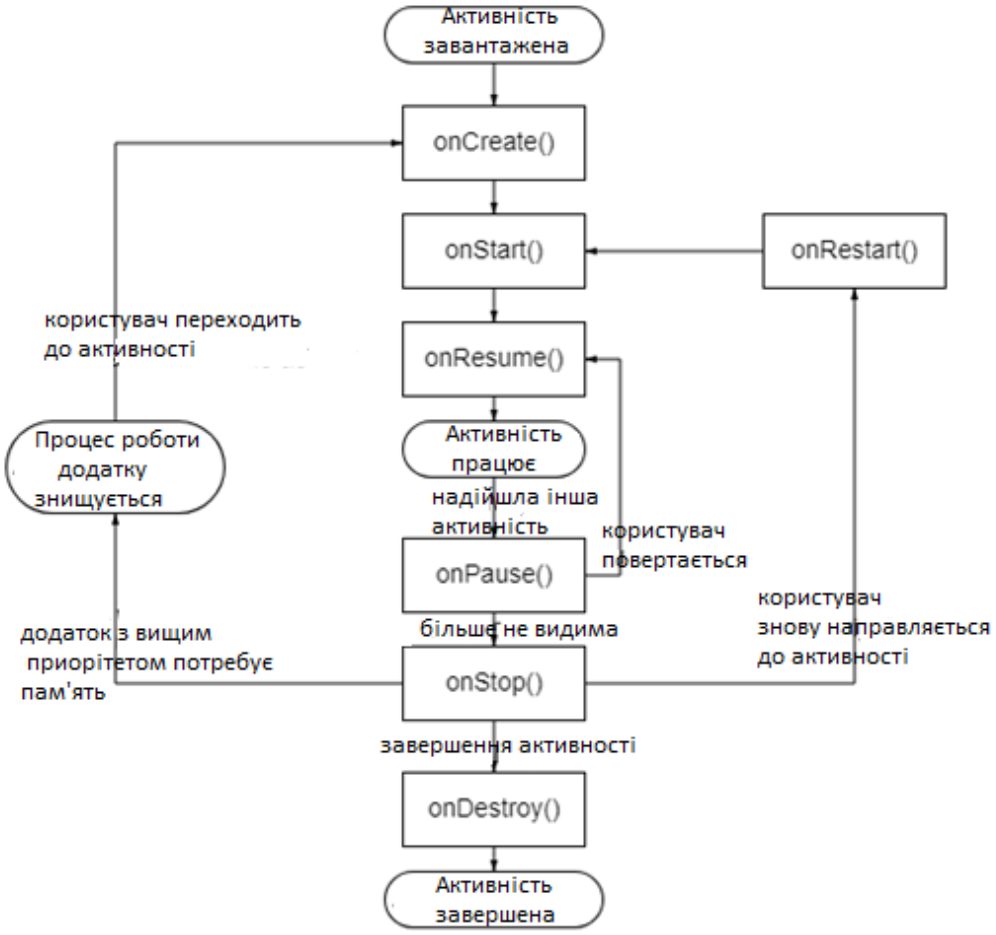

Рисунок 7 – Спрощене зображення життєвого циклу активності

2.2 Середовище розробки

Для розробки Android-додатків існує офіційне IDE (Integrated Development Environment) Android Studio, що було створене на основі IntelliJ IDEA – інтегрованого середовища розробки для багатьох мов програмування, зокрема, Java. Android Studio містить зручний редактор коду та потужні інструменти розробки для пристроїв на базі Android ОС, які підвищують можливості розробника. Таким чином, окрім великої кількості функцій, існує також можливість перегляду проєкту в реальному часі на всіх етапах розробки. Середовище розробки має вбудований емулятор, який дозволяє створювати та контролювати роботу додатку на різних пристроях в реальному часі. Наприклад, на смартфонах, планшетах або Android TV, Android Wear. Також підтримується

функція тестування на фізичному пристрої. Android Studio має велику кількість шаблонів та компонентів, підтримує інтеграцію системи контролю версій для розробки та достатній об’єм навчальної інформації для користувачів [14].

Використання потужного вбудованого емулятора призводить до обмежень, які накладаються на апаратну частину розробника, де тестується додаток. Оперативна пам’ять має бути не менше 4 ГБ, рекомендовано 8 ГБ + 2 ГБ для Android Emulator, мінімальна роздільна здатність екрану – 1280 × 800 [15]. Для підвищення продуктивності рекомендується використовувати зовнішній пристрій для тестування, обмежити кількість відкритих проєктів, використання зайвих сервісів Google Play тощо.

Android Studio використовує гнучку систему автоматичного збирання Gradle, яка покращує ефективність тестування, збирання проєкту за допомогою генерації необхідних файлів. Такі файли містять параметри конфігурації для всіх модулів проєкту – збирання верхнього рівня, специфічні параметри налаштування для кожного з модулів – збирання на рівні модуля [16].

Мови програмування, які підтримуються даним IDE – Java, Kotlin, C++.

Інтерфейс Android Studio представлено на рис.7.

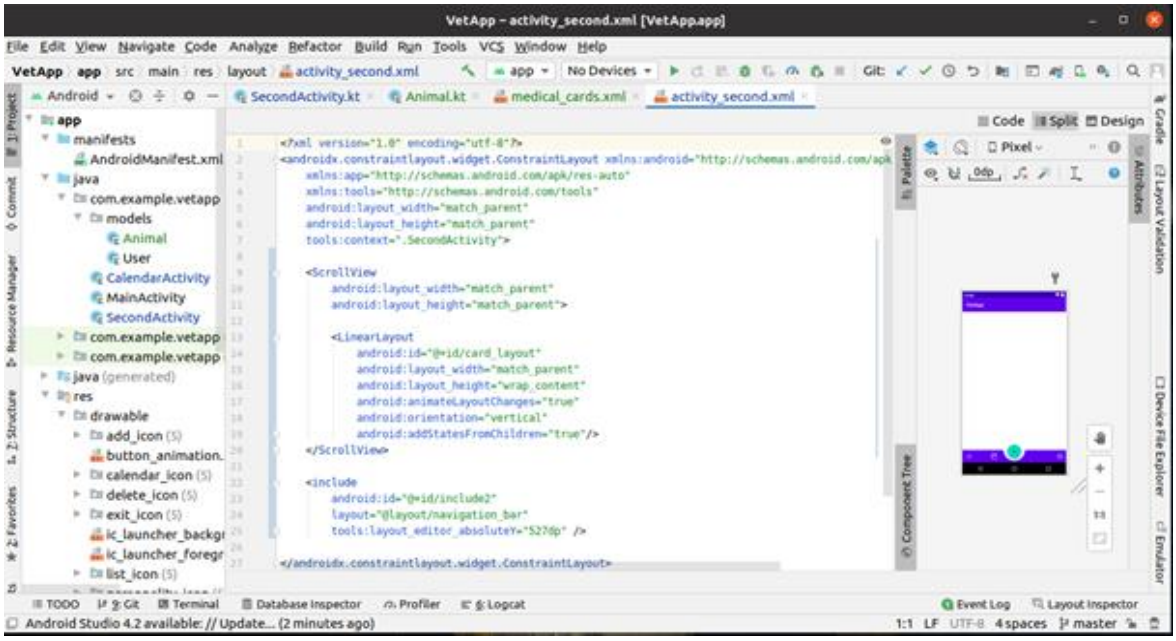

Рисунок 8 – Інтерфейс Android Studio IDE

2.3 Мова програмування

Основними мовами програмування для розробки додатків під ОС Android є Java та Kotlin. У порівнянні з Java, Kotlin є достатньо новою мовою, що була створена у 2011 році Android-розробниками від компанії JetBrains, та є Java—сумісною [17].

Мова програмування Kotlin має такі можливості та переваги:

1) компілювання в байт-код JVM;
2) здатність використовувати Java-фреймворки, бібліотеки, інтегруватися з Gradle та іншими системами збирання;
3) відкритий вихідний код;
4) конвертація Java-коду у Kotlin та навпаки;
5) достатня легкість засвоєння та зрозумілий синтаксис;
6) Null-безпечність, що попереджає компіляцію коду при спробі присвоїти або повернути null – NullPointerException, та підтримка окремих Nullable-типів.

Kotlin, на відміну від Java, забезпечений спеціальними класами для зберігання даних – Data Classes, які генерують різні шаблони, що спрощують синтаксис, зумовлюють його гнучкість. Також класи можуть розширювати свою функціональність без успадкування за допомогою функцій-розширень. Типи у мові Kotlin приводяться автоматично у більшості випадків, які не вимагають явного приведення, тому вказувати їх не обов’язково. Оскільки Kotlin підтримує функціональне програмування, існують такі особливості даного стилю, як:

1) функції вищого порядку – такі, що приймають як аргументи і повертають інші функції;
2) лямбда-вирази – анонімні функції, які передаються як вирази;
3) перевантаження операторів тощо [18].

До недоліків можна віднести більше часу, що відводиться для збирання коду, у порівнянні з Java-кодом, проте показники при частковому збиранні є кращими [15].

2.4 Засіб синхронізації даних

Альтернативним рішенням відносно традиційних СУБД виступає NoSQL – система баз даних, які зберігають дані у форматі відмінному від реляційних таблиць. NoSQL базам притаманна базова доступність, гнучкість, кінцева узгодженість, що вирішує проблему масштабованості. Схема даних в такому випадку використовує хеш-таблиці, дерева тощо [19]. Для впровадження баз NoSQL у додатки під ОС Android одним із запропонованих рішень є хмарна база даних Firebase Realtime.

Особливістю бази даних у реальному часі є доступність і забезпечення синхронізації даних між користувачами при відсутності доступу додатку. Такий автономний режим підтримується збереженням даних на диску, готових до отримання змін із хмарного сховища з відновленням доступу до мережі. Також Firebase Realtime Database сумісний з іншими інструментами Firebase – аутентифікацією, сховищем даних, засобами аналітики тощо.

Доступ до бази даних здійснюється з будь-яких пристроїв через програмний код за правилами читання та запису, описаних у правилах Firebase Realtime Database Security Rules, що визначають рівень доступу для окремих користувачів. Синтаксис правил доступу представлено на рис. 9.

```
{
  "rules": {
    ".read": true,
    ".write": true
  }
}
```

Рисунок 9 - Синтаксис правил доступу

До переваг використання баз даних в реальному часі відноситься високий рівень надійності збереження даних. Одним із шляхів досягнення безпеки є аутентифікація користувачів з підтримкою методів Google, Facebook,

анонімності тощо. Таким чином існує можливість розділення користувачів за правами доступу до окремих розділів збереженої інформації [20].

Realtime Database характеризується швидкістю проведення операцій з боку користувача та оптимальним рівнем швидкості реакції бази даних і синхронізації, що забезпечується використанням веб-сокетів. Для цього необхідно виконати відповідну структуризацію для ефективності читання та збереження даних, впорядкувати записи за допомогою ідентифікаторів.

Дані зберігаються у Realtime Database у вигляді пари «ключ-значення» згідно стандартів JSON-файлів. Використання передбачає опис запитів у програмному коді з боку додатку, що відкриває доступ до ряду зручних у використанні вбудованих функцій [21]. Кожний із записів має унікальний індекс, який використовується в якості ключа і вільний тип значення, що зумовлює відсутність строгого визначення типів. Проте запити виконуються лише за одним полем, що може суттєво зменшувати функціональність бази даних у реальному часі. Приклад структури Realtime Database представлено на рис. 10.

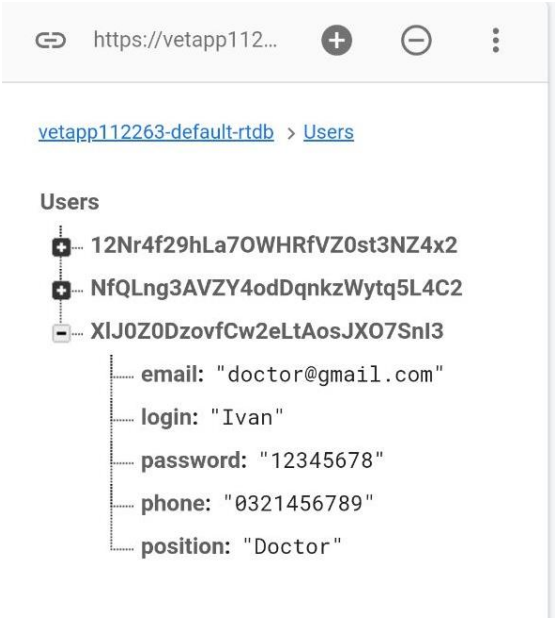

Рисунок 10 – Приклад структури Realtime Database

На противагу Realtime Database розроблено Cloud Firestore – базу даних, що має об’єктну структуру з підтримкою вкладених підколекцій або пар даних

«ключ-значення». Така ієрархія має спільне з традиційною ієрархією реляційних схем. Також у Cloud Firestore передбачені типи даних, посилання, які попереджують дублювання даних. Перевагою використання Realtime Database у порівнянні з Cloud Firestore є зручність імпорту та експорту даних у консолі Firebase для забезпечення швидкості внесення змін до набору даних без відповідних скриптів та застосування інструментів інтерфейсу командного рядка. Окрім цього у Realtime Database впроваджено відслідковування активностей і присутності користувачів у реальному часі [22]. Для того, щоб забезпечити оптимальність витрат на підтримання бази даних, необхідно врахувати особливості структури бази даних за різних потреб. Саме тому для ефективності виконання великої кількості простих запитів, швидких змін у базі даних найкращим рішенням є використання Realtime Database, а для збереження великої кількості даних із складною ієрархією рекомендовано застосування Cloud Firestore.

2.5 Обґрунтування вибору засобів розробки

В процесі вибору технологічних та програмних засобів розроблення мобільного додатку було виконано огляд та характеристику ОС Android, що доводить доцільність обрання Android у якості платформи для використання додатку. При цьому задовольняються потреби мобільності, гнучкості програмного продукту і така перевага Android, як лідерство ОС на ринку мобільних пристроїв. Розроблення додатків для ОС Android забезпечується використанням широкого набору інструментів Android SDK, які є незамінними засобами тестування, налагодження додатку в режимі реального часу.

Для розробки мобільного додатку було вибрано IDE Android Studio через потужні функціональні можливості для відлагодження програмного продукту та особливості його тестування, з огляду, на включення Android SDK, що немає аналогів. Система автоматичного збирання Gradle є безперечною перевагою

Android Studio, хоча і потребує відповідного апаратного рівня та системних витрат.

У якості мов програмування було обрано Kotlin, оскільки вона має такі переваги, як гнучкість, зрозумілість синтаксису, введення особливих безпечних типів даних, спрощення у описанні класів, Java-сумісність, що дозволяє працювати, конвертуватися у Java код. Також Kotlin інтегрується з системою збирання Gradle та використовує бібліотеки та фреймворки Java.

Для забезпечення збереження та синхронізації даних користувачів мобільного додатку було обрано Firebase Realtime Database, оскільки таке хмарне рішення дозволяє виконувати запити і отримувати на них відповідь із найбільшою швидкістю. А за умов невеликої вкладеності даних у базі, потреба у складній ієрархії зникає. Firebase Realtime Database є зручною у використанні через інструменти безпеки та можливість внесення змін і з Firebase консолі, що робить базу даних безумовно доступною для користувачів.

3 ОПИС РОЗРОБЛЕНОГО МОБІЛЬНОГО ДОДАТКУ

3.1 Структура мобільного додатку

Мобільний Android-додаток має складну структуру у зв'язку з особливостями побудови програмного продукту у IDE Android Studio. Додаток представлений єдиним модулем app, до складу якого входять:

1) файл маніфесту AndroidManifest.xml, де визначаються компоненти додатку, описані інструкції до запуску активностей, можливості доступу та взаємодії до компонентів ;

2) папка java, що містить файли з програмним кодом на мові Kotlin, розподілені у пакети додатку com.example.vetapp;

3) папка res, де зберігаються всі ресурси, що використовуються додатком.

У пакет додатку включені файли Kotlin-коду, які представляють класи для забезпечення логіки додатку та функціональності сторінок - активності, класи даних тощо.

Ресурси додатку розподілені у папці res за призначенням:

1) зображення, іконки запуску, стилі кнопок та інших елементів макету сторінки, розміщені у папці drawable;

2) файли XML макетів усіх сторінок додатку, розміщені у папці layout;

3) значення кольорів (colors), рядків (strings), тем додатку (themes) у папці values;

4) адаптивні іконки, що залежать від масштабування, зберігаються у mipmap папці.

Необхідно забезпечити незалежність ресурсів додатку через генерацію ідентифікаторів у спеціальному класі проєкту R.

Оскільки автоматизація збирання покладається на систему Gradle, у додатку присутні файли build.gradle, що містять всю необхідну інформацію про залежності та імпорт бібліотек, які будуть потрібні для збирання додатку. Існує файл збирання Gradle, який входить до складу модуля додатку, та інший, який

містить налаштування для системи. Після будь-яких змін у файлах Gradle потребується спільна синхронізація, щоб нові налаштування почали працювати.

Структуру модуля додатку представлено на рис. 11.

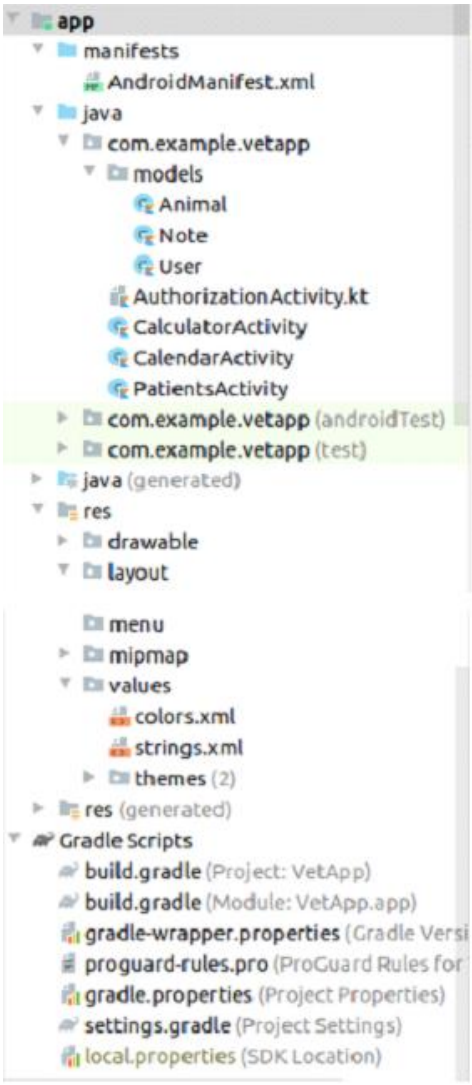

Рисунок 11 – Структура модуля

3.1.1 Графічний інтерфейс додатку

Мобільний Android-додаток складається з компонентів які відображають інтерфейс користувача на кожній із сторінок додатку – активностей. Для використання додатком будь-якого компоненту попередньо автоматично оголошується інформація про них у файлі маніфесту.

Кожна активність має власне функціональне призначення та пов’язана з іншою, що утворює систему зв’язків для забезпечення переходів користувача у додатку. Компоненти активності утворюються шляхом поєднання макету, який описує зовнішній вигляд сторінки у файлах XML з папки layout, та логіки, описаної у відповідних Kotlin-класах у com.example.vetapp.

Після запуску додатка користувачем, розпочинає життєвий цикл початкова активність AuthorizationActivity. Для відображення інтерфейсу сторінки авторизації використовується макет activity_authorization. Сторінку авторизації представлено на рис. 12.

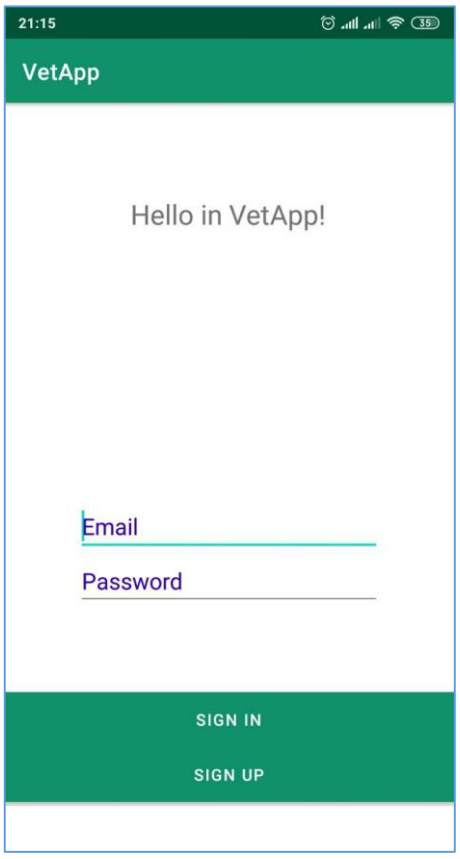

Рисунок 12 – Вигляд сторінки аторизації

Аутентифікація користувача здійснюється за електронною поштою та паролем при натисненні кнопки SIGN IN, після чого при коректному введенні даних користувач буде авторизований.

У додатку передбачена функція реєстрації нового користувача. Після натиснення кнопки SIGN UP відкривається додаткове вікно, в якому пропонується введення мінімальної інформації необхідної для реєстрації. Поля, що вимагають обов’язкового заповнення:

1) електронна пошта співробітника клініки;
2) пароль прихованого типу;
3) ім’я користувача;
4) номер особистого телефону;
5) посада працівника – доктор або асистент.

Пароль має довжину не менше, ніж 8 символів. Посада «доктор» передбачає більші можливості використання функцій додатку тоді, коли на асистента накладаються обмеження. Після успішного завершення реєстрації новий користувач може здійснювати аутентифікацію.

Вікно реєстрації представлено на рис. 13.

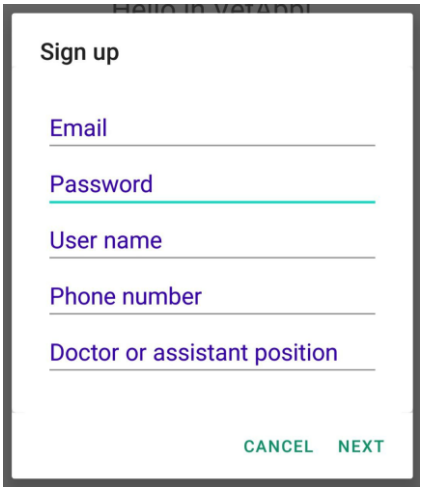

Рисунок 13 – Вигляд вікна реєстрації

Наступна активність PatientsActivity розпочинається з виведення привітання співробітника та медичних карток пацієнтів, що були раніше збережені у базі даних. Основна сторінка має інтерфейс, описаний у макеті activity_patients. Графічний вигляд карткової системи забезпечується макетом medical_cards.

Сторінку з медичними картками пацієнтів представлено на рис. 14.

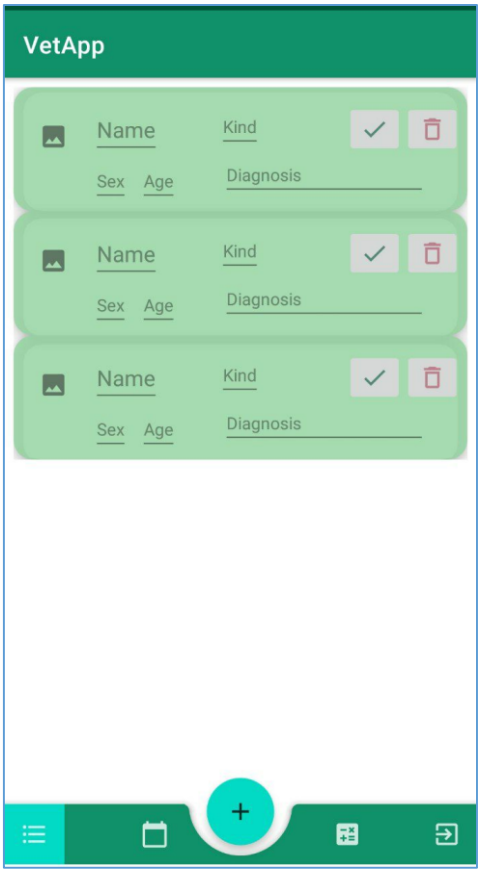

Рисунок 14 – Вигляд сторінки з медичними картками пацієнтів

Медичні картки пацієнтів складаються з полів, у які користувач може записувати дані про тварину. Початковий вигляд медичної картки має спрощений вигляд, але при натисненні на іконку відбувається розгортання інформації. Детальну інформацію надає макет detailed_record. Медична картка містить такі поля:

1) ім’я пацієнта;
2) стать та вік пацієнта;
3) вид тварини;
4) діагноз хворого;
5) показники стану, що включають:
    a. температуру тіла;
    b. серцебиття;
    c. кількість дихальних рухів;

d. діурез;
e. дефекацію;
f. апетит;
g. спрагліcть;

6) призначення лікаря, процедури та нотатки;
7) дані власника, які складаються з:
   a. повного ім’я власника;
   b. номера особистого телефону;
   c. електронної пошти.

Поля медичної картки представлено на рис. 15.

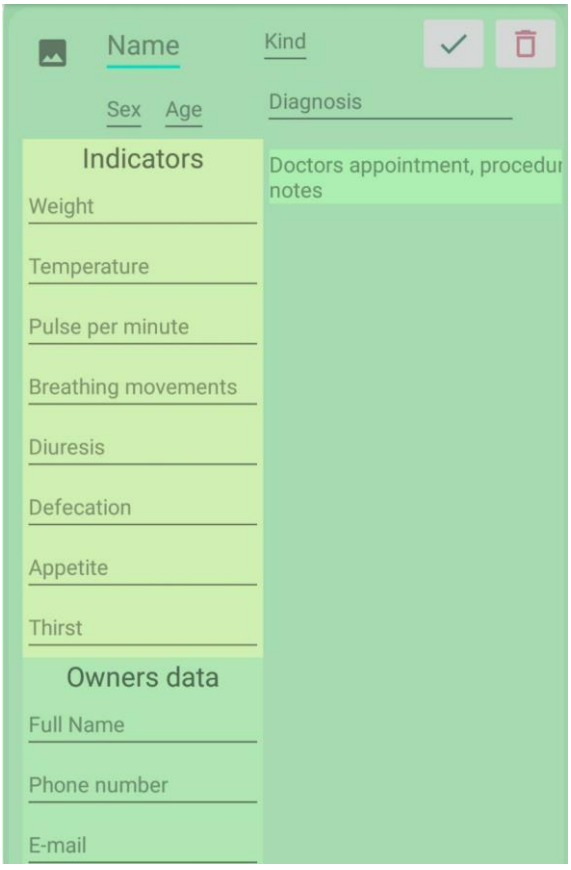

Рисунок 15 - Поля медичної картки

Передбачено такі операції з медичними картками, як створення, збереження та видалення інформації. Розподілення прав користувачів надає можливість видалення карток лише особи на посаді доктора. Після збереження даних, інформація одразу вноситься до бази даних пацієнтів.

Макет navigation_bar описує графічний вигляд навігаційної панелі, яку містять активності PatientsActivity, CalendarActivity, CalculatorActivity, та забезпечує перехід між сторінками при натисненні на відповідну іконку. Також присутня центральна кнопка навігаційної панелі «+», яка дозволяє додавати медичні картки в активності PatientsActivity та нові задачі в CalendarActivity.

Передбачено повернення з акаунту на сторінку авторизації при натисненні на відповідну іконку. Викликається вікно для встановлення остаточних намірів користувача щодо виходу, що представлено на рис. 16.

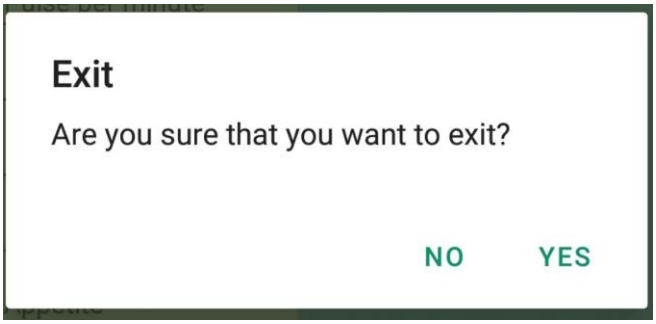

Рисунок 16 – Вигляд вікна виходу з додатку

Після переходу на сторінку планувальника відкривається інтерфейс CalendarActivity з графічним відображенням макету activity_calendar. Початково на сторінці зображено календар з поточною датою користувача. Після вибору необхідної для користувача дати зовнішній вигляд календаря змінюється, та висвітлюються задачі, що заплановані на обраний день та збережено у базу даних. Зовнішній вигляд системи записів у планувальнику описує макет todo_notes. Задачі містять такі поля:

1) статус виконання задачі;
2) запланований час задачі;
3) сутність задачі, що планується;
4) автор, який створив задачу.

Передбачено функції додавання нової задачі, при якій автоматично встановлюється ім’я автора, збереження змін у базу даних, видалення, яке може здійснювати тільки сам автор.

Сторінку планувальника представлено на рис. 17.

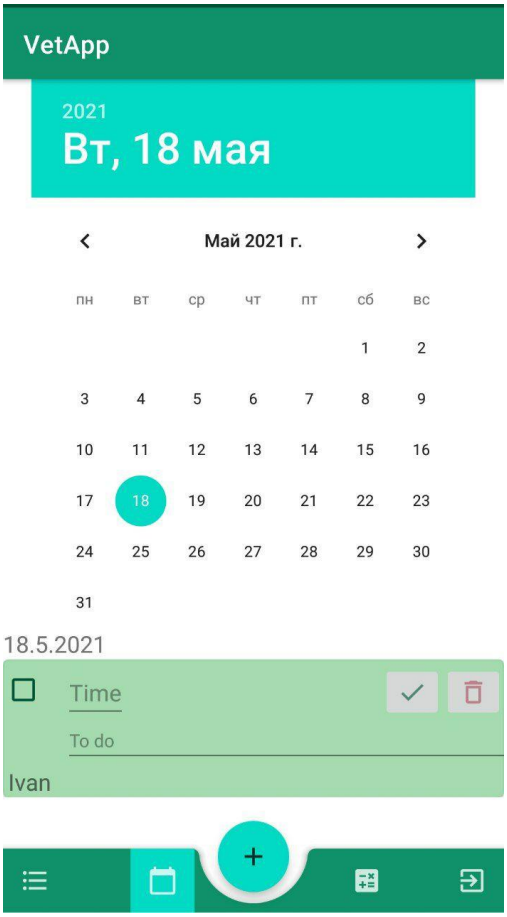

Рисунок 17 – Вигляд сторінки планувальника

У додатку передбачено калькулятор, який реалізовано активністю CalculatorActivity та макетом activity_calculator. Для зручності користування інтерфейс виконано українською мовою. Розроблений калькулятор рахує разове дозування ліків на масу тіла тварини, що зазначено у заголовку сторінки.

Для отримання результату необхідно виконати введення таких вхідних даних для розрахунку:

1) маса тіла тварини (кг);
2) кількість діючої речовини / 100 мл розчину (г);

3) доза діючої речовини / кг маси тіла тварини (г/кг).

При натисненні на кнопку «Результат» виводиться значення дозування ліків, одиниці вимірювання якого – мл.

Сторінку калькулятора представлено на рис. 18.

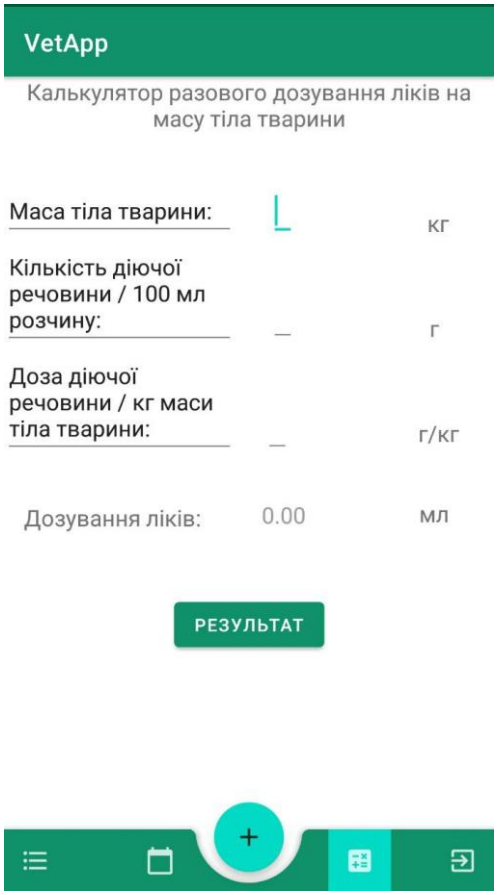

Рисунок 18 – Вигляд сторінки калькулятора

3.1.2 Структура класів додатку

У процесі розроблення додатку реалізовано структуру класів, які забезпечують функціональні можливості додатку. Додаток представлено такою структурою класів:

1) Класи активностей, що включають:
   a. AuthorizationActivity;
   b. PatientsActivity;
   c. CalendarActivity;

d. CalculatorActivity;

2) Класи даних, що зберігаються у базі даних:

a. Animal;

b. Note;

c. User.

Клас AuthorizationActivity є успадкуванням компоненту активності і вхідною точкою запуску додатку. Клас складається з таких методів:

1) onCreate;

2) signingWindow;

3) registrationWindow.

Життєвий цикл найпершої активності створюється за допомогою onCreate. Для використання інформації із бази даних виконується запит та отримання посилання на підгілку користувачів додатку. Після натиснення кнопок входу або реєстрації onCreate здійснює виклик методів signingWindow та registrationWindow відповідно.

У методі signingWindow перевіряється, чи була правильно введена електрона пошта та пароль, та виконується пошук користувача з метою аутентифікації у базі даних. У позитивному випадку створюється намір активності AuthorizationActivity здійснити перехід до активності PatientsActivity, що є наступною після входу. При цьому виконується передача інформації з бази даних, а саме, ім’я користувача та його посада. У негативному випадку виводиться інформація про помилку авторизації.

У методі registrationWindow створюється нове вікно у поточній сторінці авторизації, де при правильному введенні даних і натисненні «позитивної» кнопки здійснюється запис нового користувача до бази даних. Інакше буде виведено повідомлення про помилку. При натисненні «негативної» кнопки у методі виконується повернення до onCreate.

Клас PatientsActivity є успадкуванням компоненту активності та містить такі методи:

1) onCreate;

2) getData;
3) newRecord;
4) deleteRecord;
5) collapseExpandeTextView;
6) openCalendar;
7) openCalcul;
8) onBackPressed.

У методі onCreate створюється активність PatientsActivity, отримуються параметри наміру, переданого з AuthorizationActivity, і посилання на підгілку пацієнтів бази даних, після чого викликається метод getData.

У методі getData виводиться вміст підгілки пацієнтів бази даних у вигляді медичних карток та описується логіка, що передбачає збереження нових змін існуючого пацієнта у базу даних та видалення карток deleteRecord за умови рівня користувача «doctor» при натисненні відповідної кнопки. Таким чином, програмно визначаються параметри макету medical_cards. Викликається collapseExpandeTextView при натисненні на відповідну кнопку детальної інформації пацієнта.

У методі deleteRecord виконується пошук за іменем по базі даних пацієнта, що повинен бути видалений, та видалення.

У методі collapseExpandeTextView програмно визначаються параметри макету medical_cards, що дозволяє робити приховані елементи видимими, а видимі – навпаки.

Функціональні можливості навігаційної панелі реалізовано у методі onCreate. Прослуховувачі натиснень відслідковують дії користувача з кнопками навігаційної панелі та викликають методи створення нової медичної картки newRecord, переходу до сторінки календаря openCalendar, переходу до сторінки калькулятора openCalcul, передаючи ім’я користувача та позицію. Також реалізовано вихід через виклик метода onBackPressed.

У методі newRecord до бази даних додаються нові медичні картки, які користувач заповнює через поля, що редагуються, та зберігаються при

натисненні кнопки збереження. За допомогою виклику collapseExpandeTextView для детального перегляду, deleteRecord для видалення картки, збереження даних програмно визначаються параметри макету medical_cards.

Перехід до сторінки календаря реалізовано у методі openCalendar, де створюється намір активності PatientsActivity до CalendarActivity з параметрами імені користувача та посади. Також забезпечено вихід з активності на сторінку авторизації.

Для того, щоб перейти у нову активність CalculatorActivity, викликається метод openCalcul, де реалізовано створення наміру перейти до активності. При цьому передаються дані користувача – ім’я, посада.

У методі onBackPressed реалізовано відкриття вікна з попередженням виходу та завершення поточної активності у разі позитивної відповіді та повернення до onCreate у разі відмови.

Клас CalendarActivity складається з таких методів:

1) onCreate;
2) newTodo;
3) getNotes;
4) deleteNotes;
5) openCalcul;
6) openList;
7) onBackPressed.

У методі onCreate реалізовано створення активності, отримання параметрів наміру та посилання на гілку записів Note бази даних. За допомогою вбудованого datePicker виконується ініціалізація календаря з поточною датою. При зміні дати натисненням на календар викликається метод getNotes. Навігація забезпечується прослуховуванням натиснення на кнопки навігаційної панелі. Таким чином, при натисненні кнопки add_button виконується виклик метода newTodo.

У методі getNotes дані, отримані з бази даних задач, фільтруються за датою, що була встановлена за допомогою datePicker, і виводяться у

відсортованому за часом вигляді. Виконується програмне визначення логіки елементів макету todo_notes.

У методі newTodo виконується програмне визначення функціональних можливостей елементів макету задачі todo_notes, а саме, запис до бази даних нової задачі, видалення за допомогою виклику метода deleteNotes після перевірки поля author на рівність з параметром ім’я користувача переданого наміру.

У методі deleteNotes реалізовано пошук і видалення задачі, яка повинна бути видалена, у базі даних задач.

Для переходу на сторінку списку медичних карток пацієнтів, сторінку калькулятора розроблено метод openList та openCalcul, де створюється намір з параметрами поточного користувача – іменем та посадою, та розпочинається активність PatientsActivity або CalculatorActivity відповідно.

Для завершення активності та переходу на початкову активність авторизації створено метод onBackPressed, який викликається з onCreate та відображає вікно вибору для здійснення виходу.

Клас CalculatorActivity є наслідуванням компоненту активності та містить такі методи:

1) onCreate;
2) openList;
3) openCalendar;
4) onBackPressed.

У методі onCreate реалізовано створення аткивності, отримання наміру з параметрами імені користувача та посади для подальшої передачі у методи навігаційної панелі openList, openCalendar. Цим забезпечено створення нового наміру з параметрами для початку наступної активності при натисненні на відповідну кнопку панелі.

Для отримання вхідних для розрахунку даних у onCreate виконується прослуховування заповнення полів та введення по натисненню кнопки result. Після успішного заповнення значення обраховуються за таким рівнянням:

$$x = \frac{a \times b \times 100}{c},$$

де x – кількість разового дозування ліків тварині;

a – маса тіла тварини;

b – доза ліків на 1 кг ваги тварини;

c – кількість діючої речовини у 100 мл розчину.

У методі onBackPressed забезпечено завершення активності та вихід на сторінку авторизації.

Клас Animal необхідний для роботи з базою даних карток пацієнтів та складається з таких полів рядкового типу:

1) name – ім’я пацієнта;
2) sex – стать пацієнта;
3) age – вік пацієнта;
4) kind – вид тварини;
5) diagnosis – діагноз хворого;
6) weight - вага;
7) temperature – температура тіла;
8) pulse – кількість ударів серця на хвилину;
9) breathing – кількість дихальних рухів на хвилину;
10) diuresis – діурез;
11) defecation - дефекація;
12) appetite – апетит хворого;
13) thirst – спраглість пацієнта;
14) ownername – ім’я власника тварини;
15) phone – номер телефону власника;
16) email – електрона пошта власника;
17) appointments – призначення лікаря.

Для роботи з задачами планувальника, картками пацієнтів, авторизації та реєстрації розроблено клас User, який містить такі поля рядкового типу:

1) email – електрона пошта користувача;
2) login – ім’я користувача;
3) password - пароль;
4) phone – номер особистого телефону;
5) position – посада працівника в клініці.

Для реалізації планувальника задачі представлено класом Note, що має такі поля:

1) note – вміст задачі рядкового типу;
2) author – автор поточної задачі;
3) status – статус виконання задачі булевого типу;
4) date – дата призначення задачі;
5) time – час призначення задачі цілочисельного типу.

3.1.3 Використані інструменти та бібліотеки

У додатку реалізовано взаємодію з базою даних Firebase Realtime Database, яка в реальному часі забезпечує збереження та синхронізацію інформації про користувачів додатку, медичні картки пацієнтів ветеринарної клініки, задачі, які плануються або були виконані раніше.

У процесі розробки створено новий проект на платформі Firebase та встановлено підключення з додатком. При цьому встановлено нові залежності у файли автоматичного збирання, що дозволяє імпортування інструментів бази даних.

Для використання інструментів Firebase виконується імпорт необхідних інструментів – авторизація, база даних та посилання на неї. Посилання на FirebaseDatabase здійснено за допомогою DatabaseReference та DataSnapshot, що надає доступ до підгілок бази даних вбудованим методом child. Сортування даних бази забезпечено вбудованим методом orderByChild, який приймає назву підгілки для фільтрації.

Для зручного доступу до інформації, збереженої у базі даних, впроваджено структурний поділ на підгілки за видом даних та функціональним призначенням. Структуру бази даних у реальному часі представлено ідентифікатором додатку в якості кореня дерева, підгілками Animals, Notes, Users, що відповідають класам даних, який містить додаток – картки пацієнтів, задачі планувальника, користувачі.

Кожний запис оформлено у вигляді пари «ключ-значення» з унікальним ідентифікатором. Структуру бази даних додатку представлено на рис. 19.

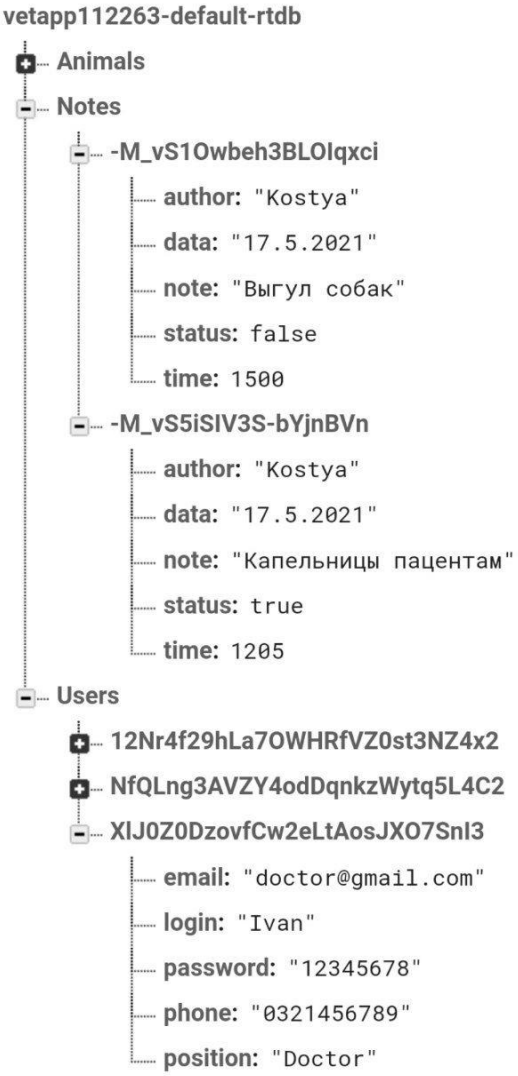

Рисунок 19 - Структура бази даних додатку

Авторизацію реалізовано через використання FirebaseAuth, що отримує посилання на гілку бази даних користувачів. У Firebase передбачено різні методи аутентифікації користувачів, а саме:

1) за електронною поштою і паролем;
2) за номером телефону;
3) засобами Google, Facebook, Twitter, Github, Microsoft, Apple тощо;
4) за анонімним входом.

У додатку реалізовано аутентифікацію за електронною поштою і паролем, що здійснюється за допомогою метода signWithEmailAndPassword. Реєстрацію користувачів забезпечено використанням подібного методу createUserWithEmailAndPassword. Візуалізацію результатів аутентифікації у додатку представлено на платформі Firebase Authentification.

Інформацію про користувачів додатку представлено на рис. 20.

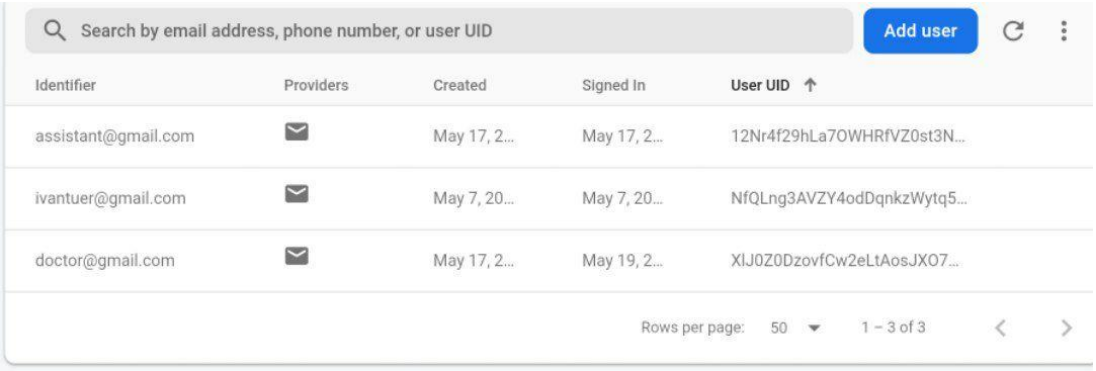

Рисунок 20 - Інформація про користувачів додатку

3.2 Опис алгоритмів

Алгоритм роботи з медичними картками зображено у на схемі додатку ІАЛЦ.045490.008 Д4. Процедура обробки включає такі елементи:

1) початок алгоритму – створення активності медичних карток пацієнтів;
2) отримання вхідних даних про користувача додатку – ім’я та посада;

3) наступний крок – встановлення зв’язку з базою даних Firebase Realtime Database карток пацієнтів ветеринарної клініки Animals;

4) виконання перевірки бази даних на наявність збережених карток. У випадку позитивного результату додатком отримується список карток, інакше – сторінка медичних карток пацієнтів залишається порожньою і виконується перевірка натиснення на кнопку add_button;

5) вивід карток отриманих з бази даних;

6) перевірка натиснення кнопки add_button, у разі чого створюється нова медична картка з порожніми полями. Якщо натиснення не зафіксовано, стан сторінки залишається незмінним, виконується перевірка натиснення наступної кнопки детального вигляду картки;

7) збереження створеної картки з порожніми полями у базу даних та перехід до чергової перевірки натиснення кнопки add_button;

8) перевірка натиснення кнопки детального вигляду медичної картки. У позитивному результаті перевірки виконується розкриття деталей, інакше – перехід до перевірки натиснення на кнопку збереження;

9) перевірка натиснення кнопки збереження. У разі натиснення на кнопку відбувається запис змін у базу даних Firebase Realtime Database, інакше – перехід до перевірки видалення картки;

10) перевірка натиснення на кнопку видалення медичної картки. У випадку натиснення виконується перехід до перевірки посади користувача, інакше – виконується перехід до перевірки натиснення кнопки навігації для переходу на сторінку планувальника ;

11) перевірка посади користувача. У разі задоволення перевірки (посада користувача – «doctor») медична картка буде видалена і зміни будуть збережені у базу даних, інакше – видалення заборонено і виконується перехід до перевірки натиснення кнопки навігації для переходу на сторінку планувальника;

12) перевірка натиснення на кнопку навігації для переходу на сторінку планувальника. У випадку натиснення виконується перехід на сторінку планувальника з перенесенням даних користувача – імені і посади та завершення

активності, інакше – перехід до перевірки натиснення кнопки переходу на сторінку калькулятора;

13) перевірка натиснення кнопки переходу на сторінку калькулятора. У випадку натиснення виконується перехід на сторінку калькулятора з перенесенням даних користувача – імені і посади та завершення активності, інакше – перехід до перевірки натиснення кнопки виходу;

14) перевірка натиснення кнопки виходу. У разі натиснення кнопки виконується перевірка намірів користувача. Якщо користувач має намір вийти, відбувається перехід у активність авторизації, поточна активність завершується, інакше відбувається перехід до перевірки натиснення кнопки додавання нової картки;

15) завершення активності – останній елемент алгоритму, за яким слідує кінець.

Спрощений алгоритм процедури обробки медичних карток наведено на рис. 21.

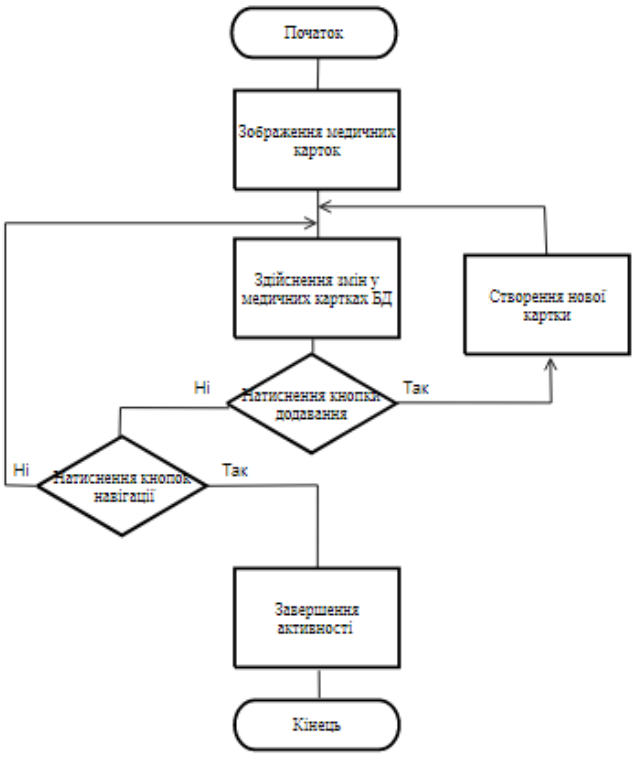

Рисунок 21 - Спрощений алгоритм процедури обробки медичних карток

Особливість алгоритму полягає у забезпеченні постійному прослуховуванні таких дій користувача, як введення нових даних у поля медичної картки, натиснення кнопок навігації та додавання нових карток.

Можливості покращення функцій додатку включають створення автоматичних нагадувань про заплановані події поточної дати, збереження фото пацієнтів для зручності ідентифікації хворих, створення окремого розділу з документацією та результатами аналізів, впровадження інших калькуляторів, впровадження доступу до персональної сторінки користувача-лікаря з контактними даними.

ВИСНОВКИ

Метою дипломного проєкту є створення мобільного Android-додатку для спеціалістів ветеринарної клініки, що, завдяки плануванню роботи спеціалістів, покращенню взаємодії між ними та оперативному відслідковуванні стану пацієнтів, дозволить підвищити ефективність їх роботи.

У ході виконання дипломного проєкту проведено аналіз предметної області, розглянуто різні за функціональним призначенням популярні додатки та доведено актуальність обраної теми дипломного проєкту.

За результатами аналізу сучасних засобів створення мобільних додатків для розробки обрано платформу Android, сучасну мову програмування Kotlin, середовище розробки Android Studio з ефективними інструментами створення додатків. Для забезпечення роботи з даними використано Firebase Realtime Database, ефективність роботи якої підтверджено результатами розробки.

Створений мобільний Android-додаток для ветеринарних спеціалістів забезпечує наступні функціональні можливості:

- здійснювати авторизацію спеціалістів ветеринарної клініки з розподілом прав доступу;
- створювати і підтримувати медичні картки пацієнтів;
- планувати процес лікування пацієнтів;
- визначати дозування лікарських препаратів;
- контролювати стан пацієнтів за показниками;
- підтримувати базу довідкової інформації;
- надає зручний інтерфейс користувача.

Розроблений мобільний додаток призначено для використання практикуючими ветеринарними спеціалістами в їх повсякденній професійній діяльності, що дозволить підвищити ефективність їх роботи в цілому.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Ветменеджер // [Електронний ресурс]. – Режим доступу: https://vetmanager.ru/ (дата звернення 20.03.2021).
2. Small Animal Emergency Medicine. Kimba Veterinary Apps // [Електронний ресурс]. – Режим доступу: https://www.kimbavetapps.com/ (дата звернення 20.03.2021).
3. Vetcalculators // [Електронний ресурс]. – Режим доступу: https://vetcalculators.com/ (дата звернення 20.03.2021).
4. ViralVet // [Електронний ресурс]. – Режим доступу: https://www.viralvet.com/ (дата звернення 20.03.2021).
5. Android Operating System Definition // [Електронний ресурс]. – Режим доступу: https://www.investopedia.com/terms/a/android-operating-system.asp (дата звернення 25.03.2021).
6. Чим відрізняються відкрита і закрита операційні системи // [Електронний ресурс]. – Режим доступу: https://ukr.fixaslowcomputer.com/3334834-what-is-the-difference-between-open-and-closed-operating-systems (дата звернення 25.03.2021).
7. Версии Android и их особенности // [Електронний ресурс]. – Режим доступу: http://f1-it.ru/versii-android-i-ih-osobennosti.html (дата звернення 25.03.2021).
8. How to install Android SDK (Software Development Kit) // [Електронний ресурс]. – Режим доступу: https://www.androidauthority.com/how-to-install-android-sdk-software-development-kit-21137/ (дата звернення 12.04.2021).
9. Android Architecture – Tutlane // [Електронний ресурс]. – Режим доступу: https://www.tutlane.com/tutorial/android/android-architecture (дата звернення 12.04.2021).

10. Download Android Studio and SDK tools // [Електронний ресурс]. – Режим доступу: http://android.com.ua/dalvik_vm_machine.html (дата звернення 12.04.2021).

11. Application Fundamentals | Android Developers // [Електронний ресурс]. – Режим доступу: https://developer.android.com/guide/components/fundamentals (дата звернення 17.04.2021).

12. Understand Activity Lifecycle | Android Developers // [Електронний ресурс]. – Режим доступу: https://developer.android.com/guide/components/activities/activity-lifecycle?hl=en (дата звернення 17.04.2021).

13. Android Studio: особенности. Достоинства и недостатки // [Електронний ресурс]. – Режим доступу: https://arduinoplus.ru/android-studio/ (дата звернення 25.04.2021).

14. Download Android Studio and SDK tools | Android Developers // [Електронний ресурс]. – Режим доступу: https://developer.android.com/sdk/index.html (дата звернення 2.05.2021)

15. The Ins and Outs of Gradle – Code – Envato Tuts+ // [Електронний ресурс]. – Режим доступу: https://code.tutsplus.com/tutorials/the-ins-and-outs-of-gradle--cms-22978 (дата звернення 2.05.2021)

16. Kotlin – конкурент Java и Scala| Открытые системы // [Електронний ресурс]. – Режим доступу: https://www.osp.ru/os/2011/07/13010422 (дата звернення 3.05.2021)

17. Функции – Kotlin // [Електронний ресурс]. – Режим доступу: https://kotlinlang.ru/docs/reference/functions.html (дата звернення 3.05.2021)

18. Kotlin vs Java: скорость компиляции // [Електронний ресурс]. – Режим доступу: https://ichi.pro/ru/kotlin-vs-java-skorost-kompilacii-253718561004381 (дата звернення 3.05.2021)

19. What is NoSQL? NoSQL Databases Explained // [Електронний ресурс]. – Режим доступу: https://www.mongodb.com/nosql-explained (дата звернення 14.05.2021).

20. Firebase Realtime Database | Security // [Електронний ресурс]. – Режим доступу: https://firebase.google.com/docs/database/security (дата звернення 14.05.2021).

21. Firebase Realtime Database // [Електронний ресурс]. – Режим доступу: https://firebase.google.com/docs/database (дата звернення 14.05.2021).

22. Choosing a Firebase Database For Your App: Realtime Database vs. Cloud Firestore // [Електронний ресурс]. – Режим доступу: https://savvyapps.com/blog/firebase-realtime-database-vs-cloud-firestore-for-your-app (дата звернення 14.05.2021).