МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра прикладних інформаційних систем

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

НА ТЕМУ

Мобільний застосунок для пошуку домашніх тварин

Галузь знань **12 «Інформаційні технології»**

Спеціальність **122 «Комп'ютерні науки»**

Освітня програма «**Прикладне програмування»**

Освітній рівень: бакалавр

Виконав: студент 4 курсу, групи ПП-42

_____Дикий М.Ю_____
(прізвище та ініціали)

Керівник _____Зосімов В.В_____
(прізвище та ініціали)
_____д.т.н, доцент._____
(науковий ступінь, звання)

Унікальність тексту 91%

Випускна кваліфікаційна робота бакалавра допущена до захисту рішенням кафедри *прикладних інформаційних систем*
Протокол № 14 від 23.05.2023р.
зав. кафедри_____Плескач В.Л._____

Київ – 2023

КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ БАКАЛАВРА

№з/п	Назва етапів кваліфікаційної роботи бакалавра	Термін виконання етапів кваліфікаційної роботи бакалавра	Відмітка про виконання
1.	Вибір теми та наукового керівника кваліфікаційної роботи бакалавра	14.10.2022	Виконано
2.	Видача завдання кваліфікаційної роботи бакалавра	24.10.2022	Виконано
3.	Настановча групова співбесіда з питань кваліфікаційної роботи бакалавра	31.10.2022	Виконано
4.	Затвердження плану кваліфікаційної роботи бакалавра	01.11.2022	Виконано
5.	Підбір та вивчення літературних та інших джерел з теми дослідження	08.11.2022	Виконано
6.	Підготовка і подання науковому керівнику першого варіанту І розділу роботи	21.12.2022	Виконано
7.	Підготовка і подання науковому керівнику першого варіанту ІІ розділу роботи	31.01.2023	Виконано
8.	Підготовка і подання науковому керівнику першого варіанту ІІІ розділу роботи	30.03.2023	Виконано
9.	Подання роботи у першому варіанті	28.04.2023	Виконано
10.	Оформлення пояснювальної записки кваліфікаційної роботи бакалавра	03.05.2023	Виконано
11.	Подання кваліфікаційної роботи бакалавра на попередній захист	**22.05.2023**	Виконано
12.	Врахування зауважень керівника і подання роботи в остаточному варіанті (з відповідним висновком про допуск) на кафедру	26.05.2023	Виконано
13.	Затвердження роботи в цілому (підготовка письмового відгуку керівника, письмова рецензія на бакалаврську роботу)	12.06.2023	Виконано
14.	Захист кваліфікаційної роботи бакалавра	28.06.2023	Виконано

Здобувач вищої освіти _____

Керівник _____
(підпис)

ВІДОМІСТЬ ДИПЛОМНОЇ РОБОТИ

Складові частини дипломної роботи	Обсяг, арк.
Титульний аркуш	1
Календарний план дипломної роботи	1
Відомість дипломної роботи	1
Анотація	1
Анотація (іноземною мовою-англійською)	1
Перелік скорочень, умовних позначень	1
Зміст	1
Вступ	2
1	8
2	26
3	31
Висновки	1
Перелік використаних джерел	3

ДП ХХХХ 00.0000.00

	ПІБ	Підпис	Дата		
Розробн	Дикий М.Ю			Лист	Листів
Керівн.	Зосімов В.В		Відомість		
Н/контр	Кравченко К.В.		дипломної		
			роботи		
Зав.каф	Плескач В.Л				

АНОТАЦІЯ

Кваліфікаційна робота: 59 с.,19 рис., 55 джерел, 2 дод.

Ця робота присвячена проектуванню та розробленню

програмної системи для адопції домашніх тварин.

Метою дипломної роботи є підвищення ефективності управління процесами адопції тварин, які перебувають у притулках.

Для досягнення поставленої мети треба вирішити такі завдання:

• дослідити загальнотеоретичні засади понять пошуку домашніх тварин та інформаційних систем спрямовані на адаптацію тварин у нових власників;

• здійснити аналіз програмно-технологічних рішень побудови інформаційних систем для пошуку домашніх тварин;

• спроектувати, реалізувати, впровадити інформаційну систему для пошуку домашніх тварин з урахуванням інженерії вимог.

Об'єктом дослідження є процеси адопції тварин за допомогою новітніх технологій.

Предметом дослідження є програмно-технічні, організаційні засади, принципи, підходи щодо побудови програмної системи адопції з використанням Android, Firebase, MVVM та мови програмування Kotlin.

Методи дослідження

Системний аналіз і синтез, моделювання, метод порівняння, що застосовано для аналізу наявних ресурсів та інформаційних систем адопції тварин.

Практичне значення полягає у тому, що розроблена програмна система може бути застосована для адопції тварин з урахуванням сучасних вимог до взаємодії з користувачами, що буде вагомою підтримкою притулків, які в умовах війни є перевантаженими та вимушені шукати шляхи др користувачів через мережу Інтернет.

Ключові слова: програмна система, адопція, Android, MVVM, Firebase.

ABSTRACT

Thesis: 59p. 19 figures, 55 sources, 2 appendices.

This thesis is devoted to the design and development of a software system for pet adoption.

The purpose of the thesis is to create a convenient and modern mobile application for the adoption of animals in shelters.

To achieve this goal, the following tasks need to be solved

- to study the general theoretical foundations of the concepts of pet search Information systems aimed at animal adoption;

- to analyze software and technological solutions for building of information systems for pet search; to analyze software and technological solutions for building

- to design, implement, and implement an information system for pet search taking into account requirements engineering.

Object of study is the process of developing an information system to support the process of animal adoption using the latest technology.

Subject of study is the software and technical, organizational principles, principles, approaches to building a software system for adoption using Android, Firebase, MVVM and the Kotlin programming language.

Research methods

Theory for the study of theoretical aspects of the processes of work of shelters in Ukraine, empirical analysis and synthesis of systems, used in the study of modern examples of implementation and construction of information systems for animal adoption, development and construction of own information system for adoption.

The practical significance lies in the fact that the developed software system can be used for animal adoption, taking into account modern requirements for user interaction, which will be a significant support for shelters that are overloaded during the war and are forced to look for ways to reach other users via the Internet.

Keywords: software system, adoption, Android, MVVM, Firebase.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

Адопція - процес відповідальності за домашнім улюбленцем, якого попередній власник покинув або відпустив до притулку.

Id- Унікальний ідентифікатор для об'єкта.

БД - база даних.

Фронтенд - презентаційна частина інформаційної й програмної системи, її користувацький інтерфейс і пов'язані з ним компоненти.

Бекенд - внутрішня частина інформаційної й програмної системи, її основний функціонал.

SDK - набір із засобів розробки, утиліт і документації, який дає програмістам змогу створювати прикладні програми за визначеною технологією або для певної платформи.

Qr-код - двовимірний штрих код який дає змогу зашифрувати в себе будь-які символи, слова, посилання та легко зчитати їх навіть за допомогою камери смартфона.

ВСТУП

У зв'язку з початком повномасштабного вторгнення з тимчасово окупованих територій емігрувала дуже велика кількість людей. І далеко не всі люди забирали своїх домашніх улюбленців коли виїхали з окупованих території єдине на що можна сподіватися що всі покинуті тварини колись потраплять у притулок з якого його зможе забрати людина завдяки новітнім інформаційним системам.

Актуальність цієї теми зумовлена насамперед тим що на цей час в Україні не має зручного і зрозумілого мобільного застосунку для того, щоб забирати тварин з притулків. А темпи заповнення притулків та кількість тварин які перебувають в небезпеці стрімко зростає.

Метою дипломної роботи є підвищення ефективності управління процесами адопції тварин, які перебувають у притулках.

Завдання дослідження

- дослідити загальнотеоретичні засади понять пошуку домашніх тварин та інформаційних систем спрямовані на адаптацію тварин у нових власників;

- здійснити аналіз програмно-технологічних рішень побудови інформаційних систем для пошуку домашніх тварин;

- спроектувати, реалізувати, впровадити інформаційну систему для пошуку домашніх тварин з урахуванням інженерії вимог.

Об’єктом дослідження є процеси адопції тварин за допомогою новітніх технологій.

Предметом дослідження є програмно-технічні, організаційні засади, принципи, підходи щодо побудови програмної системи адопції з використанням Android, Firebase, MVVM та мови програмування Kotlin.

Методи дослідження є системний аналіз і синтез, моделювання, метод порівняння, що застосовано для аналізу наявних ресурсів та інформаційних систем адопції тварин.

Практичне значення полягає у тому, що розроблена програмна система може бути застосована для адопції тварин з урахуванням сучасних вимог до взаємодії з користувачами, що буде вагомою підтримкою притулків, які в умовах війни є перевантаженими та вимушені шукати шляхи др користувачів через мережу Інтернет.

Структура роботи: Кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, розподілених на підрозділи та висновку.

РОЗДІЛ 1 ТЕОРЕТИЧНІ ОСНОВИ ПОШУКУ ТА АДОПЦІЇ ДОМАШНІХ ТВАРИН.

1.1 Основні поняття та особливості реалізації інформаційних систем для притулків

Основне завдання інформаційних систем як таких полягає в точному і своєчасному наданні інформації так само й у сфері пошуку домашніх тварин нам потрібно вкрай швидко доставляти інформацію про покинутих і знедолених домашніх улюбленців яких із початком війни стає тільки більше.

Наразі притулки для тварин переполені через велику кількість нових безхатніх тварин і тут можна виділити основні фактори ризику як для людей так власне і для тварин. А для самих тварин це створює небезпечну ситуацію через можливі нещасні випадки буцімто на дорозі чи в місті

Оскільки людині, а тим паче дитині такі тварини можуть завдати шкоди на вилиці, а також можуть бути переносниками різних груп вірусів та бактерії на своєму хутрі

І саме тут у пригоді стануть інформаційні системи оскільки завдяки їм можна автоматизувати більшість процесів до прикладу можна привести коли до притулків привозять нову тварину її має оглянути медична клініка яка закріплена за притулком також за необхідності надати весь спектр допомоги для того, щоб вилікувати або створити спеціальні умови для такого пацієнта. Після ж перевірки потрібно занести тварину і спеціальні бази даних які організовує або притулок або державна адміністрація після цього в деяких притулках які під’єднані до міжнародних систем адопції проводять чіпування після чого тварина навіть може отримати офіційний ветеринарний паспорт Також важливою частиною інформаційних систем

у функціонуванні таких притулків є статистика усіх тварин на утриманні оскільки для кожного з них потрібно розрахувати кількість корму та інших речей під час утримання.

Також для початку потрібно зазначити що такий процес взяття людиною домашнього улюбленця з притулку з подальшою його опікою називається адопцією.

Особливості ж реалізації таких систем ставлять перед їх розробниками непрості виклики з вибору, вибудовування правильної та ефективної внутрішньої архітектури.

Також кожна безпритульна тварина, яка пройшла через процедуру вилову має пройти реєстрацію на спеціальних ресурсах на основі міських адміністрацій та отримати повне медичне обстеження у ветеринарній клініці, яка закріплена за притулком.

Якщо брати реалізацію мобільних застосунків для адопції-то жодна з організації не розробила подібні застосунки єдині українські мобільні застосунки на тему тварин це «Animal ID» та «Strays ID» які будуть розглянуті далі, але жоден із них не стосується адопції та не надає кінцевому користувачу можливості забрати тварину з притулку.

Хоча судячи з веб ресурс навіть найбільших українських притулків у своїй діджиталізації вони просунулися не далеко. Оскільки навіть простий функціонал обліку тварин та породи дуже сильно відрізняється в різних облікових записах тварин навіть така проста річ як опублікувати фото тварини, яка знаходиться в них на обліку не завжди зроблена.

Це вже не кажучи про найменші притулки для тварин, які здійснюють свою основну діяльність через локальні чати або фейсбук групи та покладаються на «сарафанне радіо». Всі вищенаведені факти свідчать скоріше про не досконалу та погано спроектовано систему обліку та прийняття тварин.

1.2 Важливість інформування населення про можливості адопції

Безперечно в нашому сучасному світі з величезною кількістю компанії неможливо знати про всі підряд як, наприклад до початку написання роботи не знав і не бачив наскільки насправді велика спільнота притулків, захисників та волонтерів.

А деякі з цих притулків не мають великої реклами в інтернеті або декілька мільйонів підписників, тому вони живуть і існують у своїх власних інформаційних бульбашках, а люди зовні навіть не розуміють скільки роботи вони виконують. Власне якби люди знали більше про притулки були більш інформовані на цю тему то й переповнених притулків стало б менше.

Основну роль в інформуванні населення можуть відіграти великі або маленькі ЗМІ через їх медіаресурс, як на мене, найефективнішим способом просування таких притулків є власне великі засоби масової інформації яким під силу зробити соціальну рекламу або новинний сюжет на цю тему й кількість забраних тварин із притулку відразу виросте також з ефективних способів просування наразі можна виділити соц. мережі для деяких притулків це власне єдиний спосіб «самореклами» через соц. мережі такі організації отримують дуже швидкий й що найголовніше прямий контакт зі своєю аудиторією.

1.3 Нормативно правове регулювання у сфері захисту домашніх тварин

1.3.1 Проблеми та стан правового регулювання захисту тварин у світ

В основу законів будь-якої країни про регулювання захисту домашніх тварин лягла всесвітня декларація прав тварин основними положеннями якої є такі важливі та базисні пункти про поводження і

ставлення до тварин на основі яких кожна з країн світу організувала свою нормативно правову базу про захист тварин усі вони ставлять на меті захист та тваринного світу, а також різними способами регулюють кількість безпритульних тварин на вулицях міст, наприклад у Швейцарії коли розглядають справи про жорстоке поводження з тваринами для домашніх улюбленців є спеціальні адвокати які спеціалізуються на захищені їх прав.

У різних країнах на місцеву владу якраз і покладене дотримання і регулювання таких законів і для цього муніципалітети організовують різні програми з встановлення кількості безпритульних тварин, а також їх стерилізації У деяких країнах же безпритульні тварини взагалі не розглядаються як проблема, наприклад у Туреччині їх не заведено відловлювати чи якимось іншими способами контролювати їх популяцію навпаки влада вирішила обладнати для них невеликі місця для них комфортного та безпечного перебування також встановлює точки з кормом у парках скверах тому туриця намагається створити максимально безпечну середу для їх перебування насправді різне ставлення до тварин дуже сильно обумовлено історичними подіями різних країн і народів у деяких куточках світу взагалі змогли повністю побороти проблему безпритульних котів в основному за рахунок жорстоких законів і штрафів для населення за недбале або жорстоке поводження з тваринами за інформацією дослідження.

Всесвітнього товариства захисту тварин (WSPA) проблему безпритульних тварин подолали в таких країнах як Бельгія. Німеччина, Нідерланди та у Швейцарії у Швейцарії та Німеччині для того, щоб завести для себе будь-яку домашню тварину її потрібно зареєструвати та вести в спеціальну базу таким чином тварина і власник зв’язуються у єдиній системи й тварина не може бути просто викинути на призволяще. У США також взагалі виключається таке

поняття як тварина без власника, а їх популяцію контролюється за рахунок людей та вільного розмноження.

1.3.2 Проблеми та стан правового регулювання захисту тварин в Україні

Основне регулювання прав тварин як і в інших країнах у нас базується на всесвітній декларації прав тварин основним законом у такому регулюванні виступає Закон України про жорстоке поводження з тваринами від 2006-го року в ньому викладені основні статті декларації об'єднаних нації та розширені до їх повного функціонування.

У дві тисячі двадцять третьому році закон було переглянутий та внесено не мало правок в основному це стосувалося поводженням із безпритульними тваринами на нині окупованих територіях у рамках теми дипломної роботи нас найбільше цікавить стаття № 16 другого розділу закону в якій чітко розгорнута тема поводження з безпритульними тварина після їх виловлювання, вони мають бути простерилізовані та записані в спеціальну базу даних після чого передані до притулків або місць реабілітації тварин для передачі новому власнику.

Щодо контролю за розмноженням тварин за це на себе беруть відповідальність кабінет міністрів України якщо йдеться про диких тварин які не перебувають в агломераціях повністю або частково створених людиною За розмноженням же тварин які перебувають в агломерації міста відповідальність бере на себе місцеві органи врядування у яких достатньо інструментів припинення розмноження тварин наприкінці двадцять другого року стартувала всеукраїнська програма безкоштовної вакцинації та чіпування безпритульних тварин за підтримки Міжнародного фонду захисту тварин та Української асоціації лікарів ветеринарної медицини дрібних тварин і такі програми є вкрай важливими для українських міст та

селищ оскільки за даними міжнародного фонду захисту тварин Україна знаходиться на сьомому місці за кількістю безпритульних тварин за даними дві тисячі двадцять другого року це приблизно сім із половиною мільйонів безпритульних собак і котів.

Якщо на етапі медичної перевірки буде виявлено небезпечні захворювання то згідно з законодавством таких тварин потрібно умертвити за це несуть відповідальність ветеринарні клініки після чого подають встановлені законом документи про операцію до Української асоціації лікарів ветеринарної медицини дрібних тварин.

1.4 Огляд мобільних застосунків та веб ресурсів для пошуку домашніх тварин

На сьогоднішньому ринку мобільних застосунків для пошуку домашніх тварин ситуації є неоднозначною оскільки наразі в Україні офіційно функціонує 510 притулків багато з них не, мають навіть власного сайту і просто використовують веб ресурси інших організації для розміщення своїх тварин на тимчасовому затриманні, не кажучи вже про мобільний застосунок.

Такі організації в основному функціонують за рахунок фондів, грантів або приватних меценатів також присутні багато стерилізаційних центрів, які також здійснювати виловлювання тварин та на благодійній основі стерилізують їх і всі вони зазвичай передають тварин до притулків або організовують власні невеликі приміщення для утримання тварин майже єдиний спосіб для таких невеликих фондів знайти нового власника для тварин це великі притулки, які можуть розмістити їх оголошення на своїх веб ресурсах.

Однією з таких баз даних є «Animal id» ця громадська організація має багато різних проєктів так чи інакше пов’язаних із тваринами основним напрямом їх діяльності є чіпування тварин їх основна бізнес

модель заснована на тому щоби продавати користувачам із домашніми улюбленцями процедуру чіпування та занесення їх тварини до бази даних після чого вони видають спеціальний жетон на якому розміщено qr-код проскандувавши який кожна людина зможе перейти на веб сторінку тварини й знайти всю інформацію про неї та власника це може бути корисно для тварин які перебувають під наглядом людей, але загубилися для того щоби пересічна людина могла зв’язатися з власником.

Після реєстрації та авторизації користувача потрібно буде також додати свого домашнього улюбленця натиснувши на відповідні правила інтерфейсу після повної реєстрації в застосунку користувача буде перенаправлено на основну сторінку зображену на рисунку 1.1:

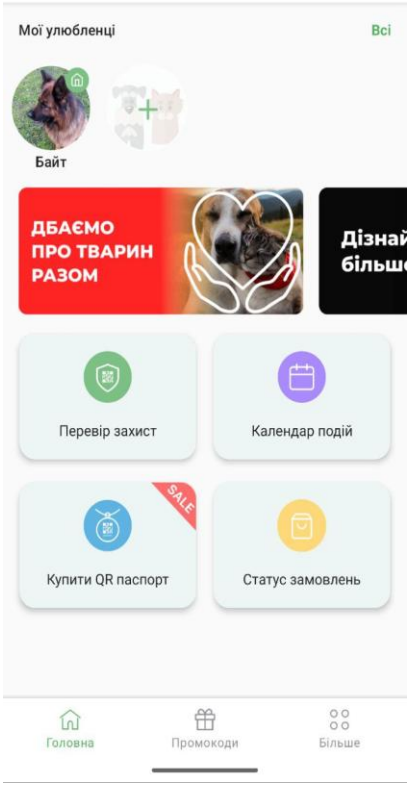

Рисунок.1.1 Головний екран Animal id

На головному екрані користувачу доступні декілька функціональних блоків такі як перевірка його улюбленця там є основна інформація яку

користувач сам надав про його улюбленця фотографії також перевірка qr паспорту тварини можливість подати її в розшук також галерея в якій користувач може переглядати фото тварини в розділі документи улюбленця можна перевірити всі наявні щеплення аналізи квитанції договори родовід та інше по суті цей розділ може слугувати користувачу як така собі медична картка тварини, бо кожен із цих документів має додати користувач натиснувши на спеціальну кнопку в інтерфейсі.Наступна з функції цієї сторінки це контроль ваги користувач також самостійно має вводити ці дані за допомогою інтерфейсу найкраща з функції на цій сторінці це історія сканування qr коду вашого улюбленця з мапою, щоб ви могли контролювати де ваша загублена тварина.

На основному екрані доступні ще чотири кнопки-функції одна з них перевірити захист на ній можна побачити всі qr коди які закріплені за вашим улюбленцем ще один з екранів це можливість для користувача запланувати події як до прикладу похід у ветеринарну клініку та робити нотатки про цей візит.

Наступна функція це купити qr паспорт як випливає з назви вона потрібна для замовлення цих ідентифікаційних жетонів на які нанесено qr код і остання функція це перевірка статусу замовлення вашого qr паспорту.

На нижній панелі навігації ми бачимо сторінку «промокоди» в моєму випадку ця сторінка залишена пустою що з погляду користувацького досвіду не дуже добре.

На сторінці ще в нас розміщенні різні пункти про застосунок, технічна підтримка та запрошення купити преміум версію застосунку з розширеним функціоналом основна функція яку надає цей преміум статус це надсилання смс коли паспорт тварини було просканово що, як на мене, не дуже гарна функція оскільки якщо ваш домашній улюбленець загубився і ви подали його в розшук ви будете перевіряти застосунку

постійно з надією що хтось знайшов вашу тварину в розділі про переваги преміум підписки також відображені такі функції як додаткові контакти для тварини й чесно кажучи не розумію чому за один додатковий запис у базі даних потрібно платити й остання перевага це онлайн підтримка 24\7.

Загалом за ціну в 40 доларів на рік вони не пропонують жодних дійсно важливих функції, але для когось це точно буде корисно.

У цілому про застосунок можна підсумувати що він зроблений якісно й дуже цілісно, а дизайн приємний і сучасний.

Оскільки я хочу зробити більш детальний огляд внутрішньої сторони застосунку я завантажив їх основний пакет встановлення та розпакував їх основний файл встановлення в програмному комплексі JD-Gui I після проведення невеликого реверс інжинірингу виявив що цім застосунком використовується вже застарілий стек технології написаний на Java судячи з внутрішніх класів для користувацького інтерфейсу вони використовують також уже трохи застарілий xml внутрішні ж сервіси застосунку написані з використання відносно нових Livedata об'єктів, а для серверної частини вони використовують тринадцяту версію NodeJs що є майже золотим стандартом в індустрії для асинхронної роботи застосунку було використано стару java реалізацію async await основною проблемою такого підходу є те що основні потоки під час виконання програми зупиняються і чекають на завершення операції цю тему я більш поглиблено розгляну в другому розділі роботи.

Спираючись на все вище викладене можна дійти висновку що на цей момент існує велика проблема в області адопції безхатніх тварин в Україні і як такого мобільного застосунку для адопції не існує, а всі веб ресурси притулків заповнені не систематизованими та не актуальними даними про тварин. На основі всіх викладених фактів вважаю за потрібне створення

сучасного, зручного та актуального для користувачів мобільного застосунку з адопції.

РОЗДІЛ 2 АНАЛІЗ ПРОГРАМНО-ТЕХНІЧНИХ РІШЕНЬ РОЗРОБКИ МОБІЛЬНИХ ЗАСТОСУНКІВ ПІД ANDROID.

2.1 Основні засади Unix систем

Для того, щоб зрозуміти що собою являє система андроїд і як розробляються застосунки під цю платформу нам потрібно розглянути кілька базових принципів юнікс подібних систем на основі яких було спочатку розроблене ядро Лінукс яке лягло в основу операційної системи андроїд.

Unix — це операційна система з відкритим вихідним кодом, Unix широко використовується різними установами завдяки своїй багатозадачності та гнучкості. Unix складається з трьох частин: ядра, командного рядка та програм. Більшість команд Unix була частково або повністю запозичена до ядра Linux. Розглянемо кожну із цих складових системи:

Shell (командний рядок)- це інтерпретатор, який обробляє команду, введену користувачем у термінал, і викликає відповідну програму з базової бібліотеки операційної системи. Командний рядок також зберігає історію команд користувача. Якщо користувач хоче переглянути історію введених команд можна скористатися стрілковими кнопками на клавіатурі або на цифровому блоку. Базовими командами для unix є cat, mv, grep, id, wc та багато інших.

Знання базових команд Unix дозволить користувачу працювати в системі Unix, підтверджувати поточний стан системи та керувати файлами

та каталогами. Базові команди Unix можна використовувати для отримання базової інформації про користувачів у вашому середовищі, навігації по файловій системі, керування файлами та дозволами на доступ, перевірки файлів та перегляду журналів.

Нижче наведено деякі з найпоширеніших і найпростіших команд Unix та їхні функції:

- ls: перелічити файли в поточному каталозі
- cd: змінити поточний каталог
- pwd: вивести в термінал поточний робочий каталог
- mkdir: створити новий каталог
- touch: створити новий файл
- cp: копіювання файлу або каталогу
- mv: перейменування або переміщення файлів і каталогів в інше місце
- rm: вилучити файл або каталог
- cat: об’єднати файли та вивести вміст у стандартний вивід
- head: показати перші 10 рядків текстового файлу (ви можете вказати довільну кількість рядків)
- tail: показати останні 10 рядків текстового файлу (можна вказати довільну кількість рядків)
- w: показувати завантаження системи, хто увійшов і що робить
- ifconfig: показати та встановити IP-адреси (зустрічається майже на всіх системах)

В Unix файлова система — це ієрархічна структура файлів і каталогів, де користувачі можуть зберігати й отримувати інформацію за допомогою файлів.

Файлова система Unix є центральним компонентом операційної системи. Вона була однією з перших частин системи. Файлова система Unix — це метод логічної організації та зберігання даних таким чином, щоб системою було легко керувати Unix все вважається файлом, включаючи фізичні пристрої, такі як DVD або USB. Початкова файлова система Unix підтримувала три типи файлів: звичайні файли, каталоги та «спеціальні файли»(файли пристроїв).

Unix використовує ієрархічну структуру файлової системи, подібну до бінарного дерева, з коренем (/) в основі файлової системи та всіма іншими каталогами, що поширюються від нього. Файлова система Unix — це по суті просто сукупність файлів і каталогів, яка має певні властивості. Вона має кореневий каталог (/), який містить інші файли й каталоги. Кожен файл або каталог однозначно ідентифікується своїм ім'ям, каталогом, у якому він знаходиться, й унікальним ідентифікатором. Кореневий каталог має номер ідентифікатора 2, а каталог lost+found має номер ідентифікатора 3. Номери ідентифікаторів файлів можна побачити, вказавши опцію -i у команді ls. Вона є самодостатньою. Не існує жодних залежностей між однією файловою системою та іншою. Каталоги мають певне призначення і зазвичай містять однакові типи інформації для легкого пошуку файлів.

Існують певні домовленості щодо розташування деяких типів файлів, таких як програми, файли конфігурації системи та домашні каталоги користувачів. Таке розташування файлів регулюється Єдиною специфікацією UNIX.

Фонд Linux (Linux Foundation). Ось так узагальнює типові місця розташування файлів в операційній системі Unix:

- /bin: Це місце, де знаходяться виконувані файли.

- /etc: Команди каталогу супервізора, конфігураційні файли, файли конфігурації диска, дійсні списки користувачів, групи, ethernet, хости, куди надсилати критичні повідомлення.
- /lib: Містить файли спільних бібліотек та іноді інші файли, пов’язані з ядром.
- /mnt: Використовується для монтування інших тимчасових файлових систем, таких як cdrom і usb.
- /proc: Містить усі процеси, позначені як файл за номером процесу або іншою інформацією, яка є динамічною для системи.
- /usr: Використовується для різних цілей і може використовуватися багатьма користувачами. Містить адміністративні команди, спільні файли, файли бібліотек тощо.
- /var: Зазвичай містить файли змінної довжини, такі як файли журналів і друку, а також будь-які інші типи файлів, які можуть містити змінну кількість даних.
- /sbin: Містить двійкові (виконувані) файли, зазвичай для системного адміністрування. Наприклад, утиліти fdisk та ifconfig.

З погляду користувача, файли організовані в деревоподібному просторі імен, де всі вузли дерева, за винятком листя, позначають імена каталогів. Вузол каталогу містить інформацію про файли та каталоги, розташовані безпосередньо під ним. Ім’я файлу або каталогу складається з послідовності довільних ASCII-символів, за винятком «/» і нульового символу «\0». Більшість файлових систем обмежують довжину назви файлу, зазвичай не більше 255 символів.

Каталог, що відповідає кореню дерева, називається кореневим. За домовленістю, його ім’я починається з косої риски (/). У межах одного

каталогу імена мають бути різними, але однакові імена можуть використовуватися в різних каталогах.

Unix пов’язує поточний робочий каталог із кожним процесом. Він належить до контексту виконання процесу і визначає каталог, який наразі використовується процесом.
Для ідентифікації конкретного файлу процес використовує ім’я шляху, яке складається з косих рисок, що чергуються з послідовністю назв каталогів, які ведуть до цього файлу. Якщо першим елементом шляху є коса риска, вважається, що шлях є абсолютним, оскільки його початковою точкою є кореневий каталог. В іншому випадку, якщо першим елементом є назва каталогу або ім’я файлу, шлях вважається відносним, оскільки його початковою точкою є поточний каталог процесу.

Операційна система Unix підтримує доступ декількох користувачів до ресурсів комп’ютера, таких як оперативна пам’ять, жорсткий диск.

Кілька користувачів можуть входити в систему з різних терміналів і виконувати різні завдання, які спільно використовують ресурси командного рядка. Це відбувається за принципом розподілу часу. Розподіл часу здійснюється за допомогою планувальника, який ділить процесорний час на кілька сегментів, які також називаються часовими інтервалами, і кожен сегмент призначається кожному користувачеві за розкладом. Коли цей час закінчується, він передається наступному користувачеві системи. Кожен користувач виконує свій набір інструкцій у межах свого часового зрізу.

2.2 Основні поняття Linux систем

Linux — це вільна операційна система з відкритим вихідним кодом, заснована на UNIX-подібному ядрі, яке вперше опублікував Лінус Торвальдс у 1991 році. Торвальдс був студентом комп’ютерних наук, який

раніше працював над UNIX OS і вважав, що вона потребує вдосконалення. Коли його пропозиції були відкинуті розробниками UNIX, він задумався про запуск ОС, яка буде сприйнятливою до змін, модифікацій, запропонованих користувачами. Ранні версії ОС Linux не були такими зручними для користувачів, оскільки їх використовували програмісти, і Лінус Торвальдс ніколи не мав на меті комерціалізувати свій продукт. Однак, завдяки відкритому вихідному коду операційна система Linux стала більш надійною та дружньою до користувачів.

Ядро Linux є основою таких популярних операційних систем, як Debian, Arch, Ubuntu та Fedora. Це одне з найпопулярніших і найпоширеніших ядер, доступних сьогодні. Linux також використовується в багатьох вбудованих системах, включаючи космічні апарати, автомобілі, телевізори, відео ігрові консолі, пристрої для розумного дому, засоби автоматизації та маршрутизатори. Linux є одним із найвидатніших прикладів співпраці відкритого та вільного програмного забезпечення. Вихідний код може розповсюджуватися, модифікуватися і використовуватися в некомерційних або комерційних цілях усіма охочими на умовах вільних ліцензій, таких як GNU GPL.

Торвальдс записався на курс Unix під час відвідування Університету Гельсінкі восени 1990-х років. На курсі використовувався мінікомп'ютер MicroVAX під управлінням Ultrix, а одним з обов’язкових текстів був «Операційні системи»: Проєктування та впровадження «Ендрюса С. Таненбаума (Andrews S. Tanenbaum). Підручник містив копію операційної системи MINIX Таненбаума. Саме завдяки цьому курсу Торвальдс уперше відкрив для себе Unix. Він зацікавився операційними системами в 1991 році.

Розчарований ліцензуванням MINIX, яке на той час обмежувало її лише освітнім використанням, він почав працювати над ядром своєї операційної системи, яке згодом стало ядром Linux.

Ядро Linux зазвичай використовується в поєднанні з проєктом GNU доктора Ричарда Столлмана. Усі сучасні дистрибутиви Linux фактично є дистрибутивами Linux/GNU. Linux є сумісною Unix-подібною ОС, яка значною мірою походить від принципів, закладених в Unix у 1970-х і 1980-х роках.

З погляду дизайну, система на основі Linux використовує ядро Linux, монолітне ядро, яке керує файловими системами, периферійним доступом, мережею та управлінням процесами. Завантажувач, такий як systemd-boot, SYSLINUX, LILO і GNU GRUB, — це програма, яка може завантажити ядро Linux в оперативну пам’ять комп’ютера, виконуючи комп’ютером після входу в прошивку і при його ввімкненні. Програма init, як і традиційна sysvinit та новіші Upstart, OpenRC і systemd, є першим процесом, оголошеним ядром Linux і коренем дерева процесів. Бібліотеки програмного забезпечення включають код, який може бути застосований запущеними процесами. Динамічний компонувальник, який керує використанням динамічних бібліотек, називається Id-linux.so у системах Linux з виконуваними файлами у форматі ELF. Загальний вигляд файлової системи зображено на рисунку 2.1. Найпоширенішою програмною бібліотекою є бібліотека GNU C (glibc) у системах Linux.

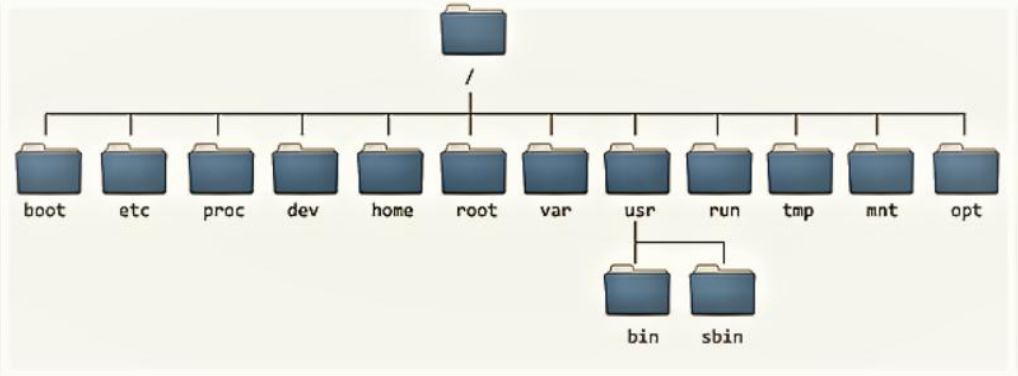

Рисунок.2.1-загальний вигляд файлової системи на основі ядра Linux.

Linux має велику базу інсталяцій кожної операційної системи загального призначення, включаючи Android, через контроль Android на базі Linux над смартфонами станом на травень 2022 року. Однак, станом на листопад 2022 року Linux використовується лише на 2,6 % настільних комп'ютерів. Впровадження Linux почалося в середині 1990-х років у спільноті суперкомп'ютерів, де такі організації, як NASA, почали все частіше змінювати свої дорогі машини недорогими кластерами комп'ютерів, що працюють під управлінням Linux. Комерційне використання почалося, коли IBM і Dell, почали надавати підтримку Linux, щоб уникнути монополії Microsoft на ринку настільних ОС. Сьогодні системи Linux повністю використовуються в обчислювальній техніці, від вбудованих систем до практично кожного суперкомп'ютера, і забезпечили собі місце в серверних інсталяціях.

2.3 Система Android.

Android — це мобільна операційна система, розроблена компанією Google. Історія Android почалася у 2003 році, коли в Пало-Альто, штат Каліфорнія, Енді Рубін, Річ Майнер, Кріс Вайт і Нік Сірс заснували компанію Android Inc. У серпні 2005 року компанія Google придбала Android Inc. і з того часу Android є дочірньою компанією Google Inc.

Цикл розробки Android характеризується швидкою ітерацією та постійним вдосконаленням. Протягом першого року комерційного існування Android компанія Google випустила нову версію кожні два з половиною місяці. Останнім часом Android працює за шестимісячним циклом розробки, а це повільніше, ніж раніше.

Існування Android було розмитим змінами. Було продано майже мільярд пристроїв і 1,5 мільйона пристроїв активуються щодня.

Наразі кодові назви Android варіюються від A до J, такі як Aestro, Blender, Cupcake, Donut, Eclair, Froyo, Gingerbread, Honeycomb, Ice Cream Sandwich, Jelly Bean, KitKat і Lollipop.

Android 0.5, Milestone 3, був першим публічним випуском Android, позначеним як «m3-rc20a», де «m3» означає «Milestone 3». Хоча Google, можливо, не оприлюднив номер версії, користувацький агент браузера ідентифікує її як «Android 0.5». Емулятор використовував qwerty-bar з дисплеєм 320x240, що відтворює реальний прототип пристрою. Пристрій було створено компанією HTC, і, здається, саме цей пристрій мав кодову назву «Sooner» згідно з багатьма ранніми повідомленнями про Android. Але Sooner так і не був випущений на ринок.

Багатозадачність і фонові програми вже працювали в Milestone 3. Вихід із програми не призводить до її закриття — програми зберігали стан, навіть текст, залишений у текстовому полі. Це була функція, яку iOS не змогла реалізувати до випуску iOS 4 у 2010 році. Попри відсутність іконки номеронабирача, емулятор Milestone 3 мав можливість здійснювати телефонні дзвінки. Карти не мали жодної інтеграції з GPS.

Меню історії сповіщень Android показує сповіщення в хронологічному порядку, з останніми вгорі. Однак система не пропонує можливості розмістити ярлик історії сповіщень на головному екрані для швидкого доступу до нього.

Файлова система Android — це структура, у якій зберігаються всі файли й дані на пристрої Android. Ось кілька ключових моментів, які потрібно знати про файлову систему Android:

Структура файлової системи Android побудована поверх ядра Linux, і вона має деякі спільні риси з файловими системами Linux яка була розглянута вище.

У файловій системі Android відсутня концепція дисків або каталогів, як у Windows. Замість цього вона використовує розділи в одному каталозі, як одне дерево з декількома гілками. Певні розділи файлової системи можна досліджувати за допомогою файлових менеджерів, що дозволяє навести певну подобу порядку у ваших файлах.

Зазвичай на кожному пристрої Android є шість основних розділів: /boot, /system, /recovery, /data, /cache та /misc. Крім того, для зовнішніх Android-смартфонів існують два розділи файлової системи — /sdcard та /sd-ext.

Кожен розділ також має певну функцію. Наприклад, розділ /system містить файли операційної системи, розділ /data — призначені для користувача дані та програми, розділ /cache — тимчасові файли, створені програмами, і так далі.

Android поділяє своє сховище на три окремі категорії, що суттєво відрізняється від структури файлової системи ПК. Це сховище пристрою, портативна SD-карта й кореневе сховище пристрою.

Без спеціальних прав користувача пристрою користувачі Android зазвичай мають доступ лише до розділу даних, який містить створені користувачем дані та програми.

Деякі Android не мають файлового менеджера, але зазвичай кожен із вендорів встановлює свій застосунок для взаємодії з файловою системою як приклад можна привести.

Файли від Google — це надійна програма для очищення пристрою, яка допомагає знаходити файли, звільняти місце та створювати резервні копії файлів у хмарі. Він сканує файли, щоби побачити, скільки вільного місця залишилося на телефоні та SD-карті.

2.4 Сучасні підходи до реалізації Android застосунків

Сучасний технологічний стек Android — це по суті набір технологій, які використовуються для розробки якісних та надійних мобільних застосунків. Технологічний стек Android містить в собі комбінацію мов програмування, бібліотек, фреймворків, серверів, UI/UX рішень, програмного забезпечення та інструментів, що використовуються розробниками.

Щоби побудувати сучасний стек технологій Android, розробники повинні враховувати як фронтенд, так і бекенд розробку. Для фронтенд-розробки існує безліч інструментів і платформ, які допомагають розробити інтерфейс мобільного застосунку. Основні інструменти, що використовуються для фронтенд-розробки, включають Kotlin, React Native, Ionic та Xamarin. Фронтенд-розробник повинен враховувати доступність і продуктивність під час розробки будь-якого проекту. Важливо, щоб застосунок виглядав незмінно чудово на пристроях будь-якого розміру та роздільної здатності, а цілі продуктивності в першу чергу пов’язані із часом рендерингу, маніпуляціями з HTML, CSS та JavaScript, щоб зменшити час завантаження.

Розробка бекенду також є критично важливою частиною розробки мобільних застосунків , яка відповідає за зберігання даних, безпеку та бізнес-логіку. Внутрішня частина мобільного застосунку схожа на сервер для мобільних застосунків, оскільки вона зберігає і сортує всю важливу інформацію, яку не бачать кінцеві користувачі. Основні інструменти розробки бекенду, необхідні для мобільних застосунків, включають мову та фреймворки, веб-сервери, системи управління базами даних, платформи мікросервісів, локальні середовища розробки, сервіси для спільної роботи та тестери продуктивності.

Найкращий стек технологій для розробки мобільних застосунків на Android охоплює Android Studio та Android Developer Tools як інструментарій. Android Studio популярний серед розробників застосунків для Android, оскільки дає свободу зосередитися на створенні унікальних і якісних застосунків. Android Developer Tools (ADT) надає повну підтримку для розробки застосунків для Android. Компоненти або інструменти, що входять до складу Android SDK, можна завантажити окремо, а сторонні доповнення легко доступні для завантаження.

Вибір найкращого технологічного стека для розробки мобільних застосунків має важливе значення для того, щоб мобільний застосунку був професійним, функціональним, масштабованим, підтримуваним і безпечним. Існує кілька важливих моментів, які слід враховувати при виборі найкращого технологічного стека для розробки мобільних застосунків, включаючи цілі програми, компанію, яка розробила технологію, можливість врегулювати на різних платформах, питання безпеки та сумісність із технологіями, які вже використовуються.

Отже, сучасний технологічний стек Android містить набір технологій, які розробники використовують для створення якісних та надійних мобільних застосунків. Фронтенд і бекенд розробка є критично важливими частинами розробки мобільних застосунків, і розробники повинні враховувати їх обидві при створенні сучасного технологічного стека Android. Найкращий стек технологій для розробки мобільних застосунків на Android містить Android Studio та Android Developer Tools в якості інструментарію. Вибір найкращого технологічного стека для розробки мобільних застосунків має важливе значення для того, щоб ваш мобільний застосунку був професійним, функціональним, масштабованим, підтримуваним і безпечним.

Як показує практика та статистика google play(основна платформа для розміщення андроїд застосунків) основною мовою програмування для нових програмних рішень є Kotlin Java на цю мить потрохи втрачає свої позиції та використовується для додавання нового функціоналу до старих уже наявних застосунків тому для написання сучасного застосунку зручніше буде використовувати kotlin.

Але для початку розберемося що це таке та із чим його їдять

Kotlin — це сучасна мова програмування загального призначення зі статичною типізацією, яка була випущена у 2016 році компанією JetBrains. Це кроссплатформенная мова, яку можна використовувати на різних платформах, таких як Windows, Mac, Linux, Raspberry Pi тощо. Kotlin повністю сумісна з Java, що означає, що код і бібліотеки Java можна використовувати в програмах на Kotlin.

Kotlin легко вивчити, особливо якщо ви вже знаєте Java. Вона була розроблена прагматично, що означає, що це мова програмування, корисна для повсякденної розробки, яка допомагає користувачам виконувати роботу завдяки своїм можливостям та інструментам. Котлін надихався багатьма мовами програмування, включаючи Java, Scala, C# та Groovy.

Котлін є статично типізованою мовою, що означає, що тип кожної змінної та виразу відомий під час компіляції. Хоча це статично типізована мова, вона не вимагає від вас явно вказувати тип кожної змінної, яку ви оголосили. Котлін має стислий синтаксис і надає багато можливостей для скорочення шаблонного коду, таких як класи даних, функції розширення тощо.

Він є об'єктно орієнтованою мовою, яка має багато функціональних елементів програмування. З об'єктно орієнтованого боку, вона підтримує номінальні підтипи з обмеженим параметричним поліморфізмом. З боку функціонального програмування вона має першокласну підтримку функцій

вищого порядку та лямбда-літералів. Котлін також має систему типів, яка розрізняє nullable та non-nullable типи, досягаючи нульової безпеки, тобто гарантуючи відсутність помилок під час виконання, спричинених відсутністю значення (тобто нульовим значенням).

Kotlin спонсорується компанією Google і була оголошена однією з офіційних мов для розробки під Android у 2017 році. Kotlin стає все більш популярною серед розробників Android, оскільки вона є більш лаконічною, виразною та безпечнішою, ніж Java. Kotlin також працює краще, ніж Java для розробки Android завдяки використанню незмінності та властивостей, а також 100 % сумісності з Java.

Наступною важливою частиною є архітектура застосунку

Існують різні типи архітектури для Android-застосунків, і кожна з них має свої переваги та недоліки. Розглянемо деякі з найпоширеніших архітектури для Android-застосунків:

MVVM (Model-View-ViewModel): Це рекомендована Google архітектура для розробки застосунків для Android. Вона відокремлює інтерфейс користувача від бізнес-логіки та рівня даних, що полегшує підтримку та тестування програми. MVVM використовує такі речі, як LiveData та ViewModels, щоб розв'язувати найпоширеніші проблеми, з якими стикаються застосунки для Android: життєвий цикл та пастки, пов’язані зі зміною ротації. Належне розділення логіки та поведінки дозволяє застосунком бути гнучкими та простими в обслуговуванні.

Загальний вигляд моделі взаємодії MVVM архітектури зображено на рисунку 2.2.

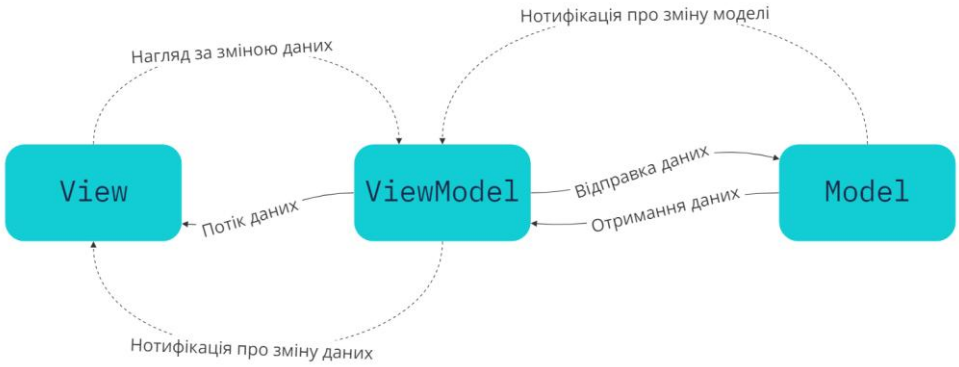

Рисунок.2.2 MVVM архітектура в блоковому вигляді

Чиста архітектура: Ця архітектура фокусується на розділенні проблем і модульності. Вона спрямована на створення коду, який легко читати, тестувати та підтримувати. Чиста архітектура розділяє застосунок на шари, кожен із яких має свою власну відповідальність: Представлення, Домен та Дані. Це може бути корисно, коли над застосунком працює кілька розробників одночасно.На рисунку 2.3 зображено загальний вигляд шарів чистої архітектури.

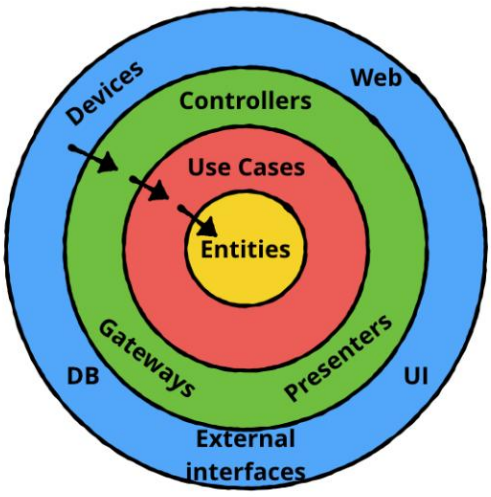

Рисунок.2.3 Загальний вигляд чистої архітектури

Патерн «Спостерігач»: Патерн «Спостерігач» визначає залежність між об’єктами типу «один-до-багатьох». Коли один об’єкт змінює стан, його залежні об’єкти отримують сповіщення та автоматично оновлюються. Цей патерн є універсальним і може використовуватися для операцій із невизначеним часом виконання, таких як виклики API, або для реагування на введення користувачем. Спочатку він був популяризований фреймворком RxAndroid, але Android також представив власний спосіб реалізації цього патерну через Live Data. Загальний вигляд патерну зображено на рисунку 2.4.

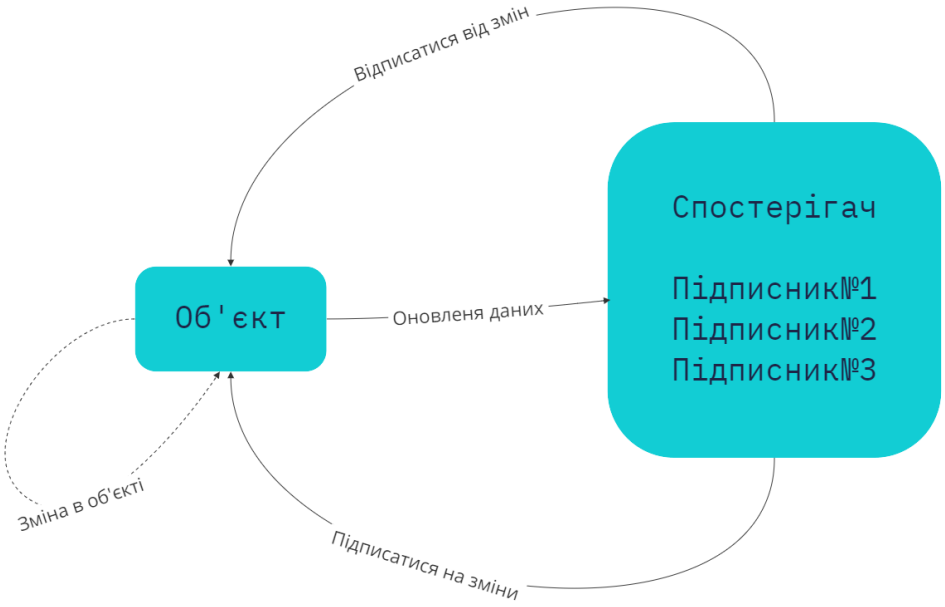

Рисунок.2.4 Загальний вигляд патерн спостерігач

Model-View-Controller (MVC): Ця архітектура розділяє застосунок на три частини: Модель, Представлення та Контролер. Модель представляє дані та бізнес-логіку, представлення — інтерфейс користувача, а контролер виступає посередником між ними. Таку архітектуру легко зрозуміти й реалізувати, але вона може призвести до тісного зв’язку між контролером і представленням, що може зробити застосунок складнішим в обслуговуванні. На рисунку 2.5 представлений загальний вигляд MVC архітектури.

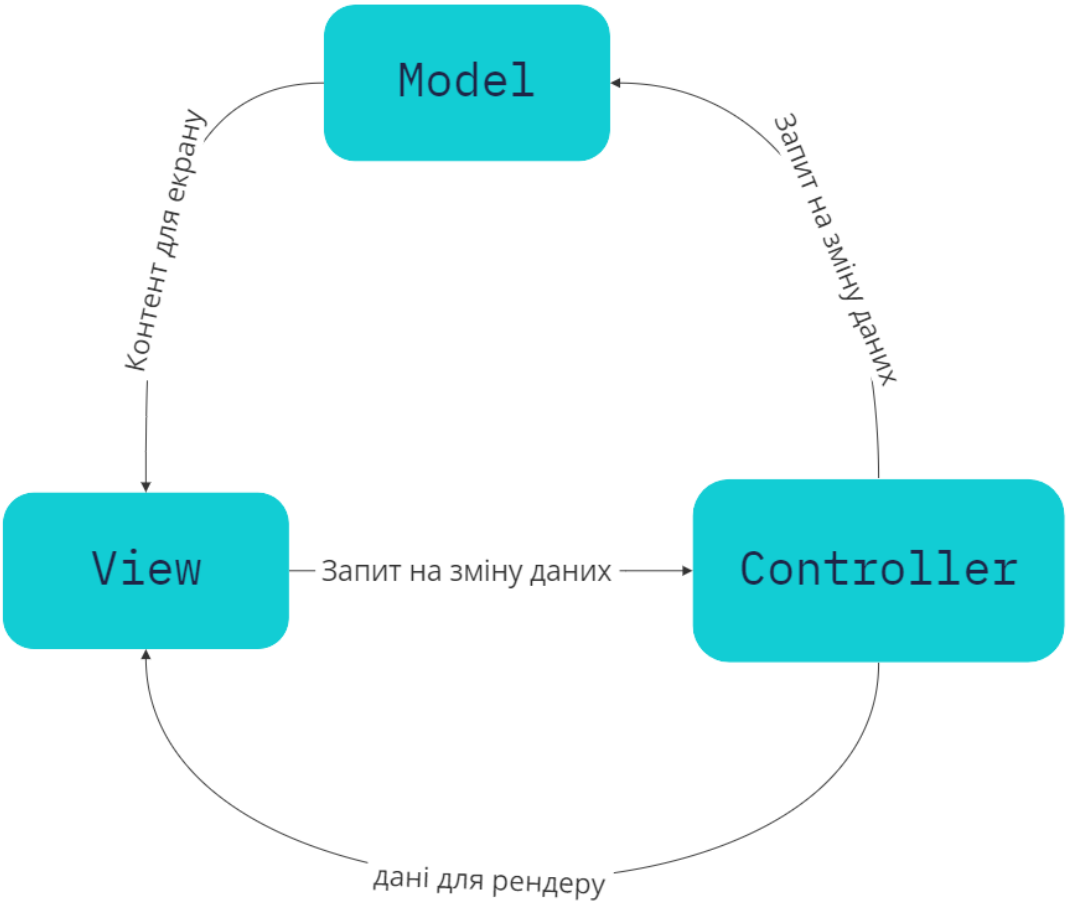

Рисунок. 2.5 MVC архітектура в блоковому вигляді

MVP Ця архітектура схожа на MVC, але вона додає шар презентатора між представленням і моделлю. Презентатор діє як посередник між ними, обробляючи вхідні дані користувача та оновлюючи представлення. MVP краще тестується, ніж MVC, але він також може призвести до тісного зв'язку між презентером і представленням.Рисунок 2.6 зображує mvp архітектуру у блоковому вигляді.

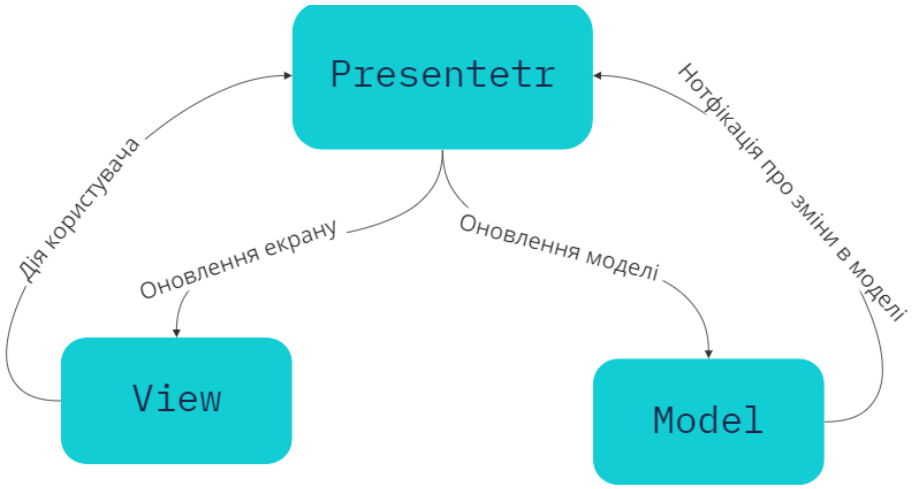

Рисунок.2.6 MVP архітектура в блоковому вигляді

Model-View-Intent (MVI): Ця архітектура є варіацією MVVM, яка підкреслює односпрямований потік даних і незмінність. Модель представляє стан застосунку, Подання відображає стан, а Intent представляє дії користувача. View надсилає Наміри до ViewModel, яка оновлює Модель і надсилає новий стан назад до View.Взаємодії всередині архітектури представлені на рисунку 2.7.

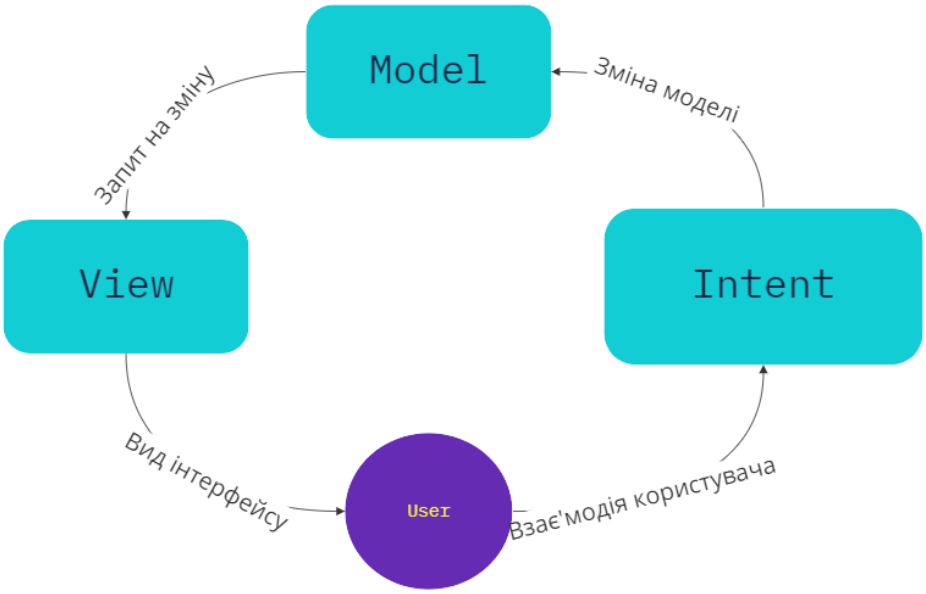

Рисунок.2.7 MVI архітектура в блоковому вигляді

Існує багато архітектур для розробки застосунків для Android, кожна з яких має свої переваги та недоліки. Рекомендованою архітектурою від Google є MVVM, яка відокремлює інтерфейс користувача від бізнес-логіки та рівня даних, що полегшує підтримку та тестування застосунку. Clean Architecture — ще один варіант, який фокусується на розділенні проблем і модульності. Observer Pattern, MVC, MVP та MVI — це інші архітектури, які можна використовувати залежно від конкретних вимог проєкт.

Також важливою частиною будь-якого проекту є бібліотеки які використовуються для впровадження певному функціоналу без потреби самостійно його писати оскільки кожна бібліотека надає певний набір функції для реалізації того чи іншого функціоналу й таких бібліотек існує дуже велика кількість, але деякі з них уже стали стандартом для індустрії мобільної розробки наступна частина розділу буде присвячена короткому огляду саме таких бібліотек.

Бібліотека Android Jetpack: Ця бібліотека є набором компонентів та інструментів, які допомагають розробникам створювати високоякісні програми для Android. Вона включає компонент навігації, WorkManager, LiveData, Paging, ViewModel та інші бібліотеки, які допомагають розробникам писати чистий, тестований і підтримуваний код.

Бібліотека Retrofit: Ця бібліотека використовується для виконання мережевих запитів і є безпечним HTTP-клієнтом для Android. Вона полегшує споживання RESTful веб сервісів, переводячи API в Kotlin дата класи або Java-інтерфейси залежно від того яку мову програмування було обрано для реалізації проєкт.

Бібліотека Coil: Ця бібліотека використовується для завантаження та кешування зображень в Android-застосунках. Вона швидка, ефективна й підтримує отримання, декодування та відображення відео, зображень та анімованих GIF-файлів.

Бібліотека Gson: Ця бібліотека використовується для серіалізації та де серіалізації об’єктів Kotlin в JSON та з JSON. Вона надає простий API для перетворення об’єктів Kotlin в JSON і JSON в Kotlin дата класи.

Бібліотека Dagger-Hilt: Ця бібліотека використовується для ін’єкції залежностей в Android-застосунках. Вона допомагає розробникам писати модульний і тестований код, надаючи можливість впроваджувати залежності в об’єкти.

На застосунку до цих ключових бібліотек, є ще кілька важливих бібліотек, які розробники можуть використовувати для створення сучасних Android-застосунків. До них відносяться

Бібліотека Room: Ця бібліотека використовується для роботи з базами даних SQLite в Android-застосунках. Вона забезпечує рівень абстракції над SQLite, що дозволяє легко працювати з базами даних об'єктно орієнтованим способом.

Бібліотека Coroutines: Ця бібліотека використовується для написання асинхронного та не блокуючого коду в Android застосунках. Вона забезпечує спосіб написання коду, який легко читати, писати та підтримувати.

Бібліотека lifecycle: Ця бібліотека використовується для управління життєвим циклом компонентів Android, таких як дії на екранах. Вона надає можливість обробляти зміни конфігурації, керувати витоками пам’яті та уникати збоїв.

Компонент Compose Navigation: Цей компонент є частиною бібліотеки Android Jetpack і використовується для керування навігацією між екранами в застосунках Android. Він надає можливість керувати навігацією в послідовний та передбачуваний спосіб.

Бібліотека AndroidX: Ця бібліотека є розширенням бібліотеки Android Support Library і використовується для забезпечення зворотної сумісності з застосунками Android. Вона включає кілька бібліотек, які допомагають розробникам використовувати нові функції Android на старих версіях Android.

Щоб використовувати ці бібліотеки, розробники повинні додати їх до файлу build.gradle свого проекту, а потім імпортувати їх у свій код. Бібліотеки доступні на репозиторії Maven Central, тому розробники можуть легко завантажити їх і додати до свого проєкт.

Ще однією важливою частиною застосунків є їх back-end частина це сервер який приймає запити від застосунку обробляє інформацію та відправляє дані у відповідь після їх обробки.

Є велика кількість різних способів реалізації back-end частини. Для свого проекту я обрав одне з рішень яке рекомендує використовувати Google для написання не дуже навантажених застосунків:

Firebase — це платформа, яка надає декілька інструментів та сервісів для створення сучасних мобільних та веб застосунків. Вона включає такі сервіси, як Firebase Realtime Database, Firebase Cloud Messaging, Firebase Authentication та Firebase Cloud Functions. Ось деякі ключові компоненти Firebase, які розробники можуть використовувати для створення сучасних застосунків для Android:

Firebase Cloud Firestore: Це база даних на основі документів NoSQL, яка дозволяє розробникам зберігати й синхронізувати дані в режимі реального часу на різних пристроях і платформах. Вона надає такі функції, як автономна синхронізація даних, оновлення даних у режимі реального часу та автоматичне масштабування. Firebase Cloud Firestore можна інтегрувати з Firebase Cloud Functions для створення без серверних

інтерфейсів API, які можна використовувати для виконання складних операцій із даними.

Firebase Cloud Functions: За допомогою Firebase Cloud Functions розробники можуть розгортати код Node.js для обробки подій, викликаних змінами в базі даних Cloud Firestore. Це дозволяє їм легко додавати серверну функціональність у свій застосунок без запуску власних серверів.

Хмарне сховище Firebase: Це хмарний сервіс зберігання об’єктів, який дозволяє розробникам зберігати та обслуговувати створений користувачем контент, наприклад, зображення, відео та аудіо файли. Firebase Cloud Storage надає такі функції, як автоматичне масштабування, безпека даних та реплікація між регіонами. Його можна інтегрувати з Firebase Authentication для контролю доступу до збережених даних.

На завершення, Firebase надає декілька інструментів та сервісів, які розробники можуть використовувати для створення сучасних застосунків для Android. Firebase Cloud Firestore, Firebase Cloud Functions та Firebase Cloud Storage є одними з ключових компонентів, які розробники можуть використовувати для створення масштабованих, без серверних API для своїх Android-застосунків, що працюють у режимі реального часу.

Висновки

У даному розділі ми в основному приділили увагу розгляду систем які лягли в основу реалізації Android розглянули причини та основи виникнення таких систем також охопили їх базові поняття після чого перейшли до власне системи Android розібралися у відмінності цих систем та розглянули що система Android запозичила з них.

У другій частині розділу розглянули основні підходи до реалізації застосунків на базі Android та розібралися із сучасним підходом до реалізації всіх частин застосунку як на front-end частині, так і на backend.

Як висновок я обрав для себе стек технології на основі яких буде розроблено практичну частину кваліфікаційної роботи.

РОЗДІЛ 3 ПРОЕКТУВАННЯ, РЕАЛІЗАЦІЯ ТА ВПРОВАДЖЕННЯ ЗАСТОСУНКУ ДЛЯ АДОПЦІЇ ДОМАШНІХ ТВАРИН.

Для реалізації практичної частини кваліфікаційної роботи нам потрібно розробити та реалізувати мобільний застосунок для пошуку домашніх тварин. Оскільки як було сказано в першому розділі аналогічних застосунків для притулків немає. Мені потрібно буде розробити застосунок спираючись тільки на власне бачення для такого типу застосунків і потреб користувача.

Основним технологічним стеком було обрано найпередовіший стек технології для розробки під Android.

Складемо приблизну діаграму функціонування застосунку та відобразимо її на рисунку 3.1:

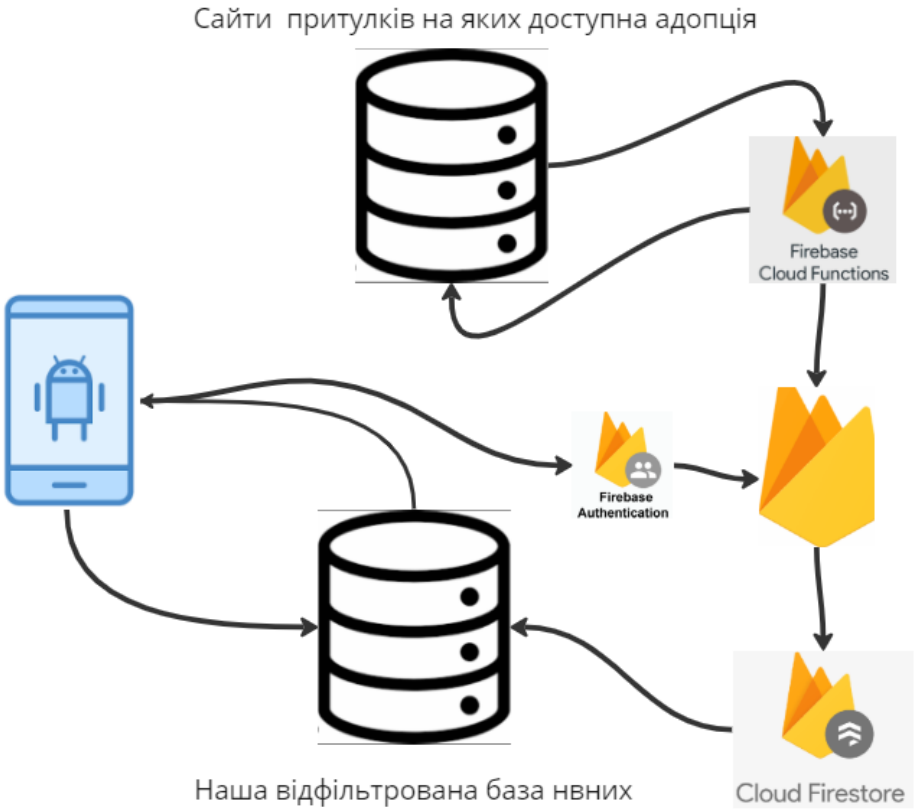

Рисунок.3.1 Приблизна діаграма функціонування застосунку

Як видно з малюнку вище основою даних для застосунку стануть вже наявні сайти притулків на яких є адопція тварин оскільки їх дані є неповними частково не очищеними та мають на впорядковані запаси для початку роботи над застосунком нам потрібно буде очистити їх переконавшись що у кожного об’єкта у нашій базі даних буде декілька картинок, ім’я, та правильно оформлений опис. Після очищення даних ми отримаємо власну базу даних яка буде оновлюватися і забирати та очищати дані раз на день.Власне з цієї підготовленої бази ми та будемо брати дані для застосунку використавши Firebase SDK.

3.1 Огляд бекенд частини застосунку

В нашому випадку бекенд частина буде мати декілька основних обов’язків такі як очищення та систематизація даних, створення нової бази з очищених даних та авторизація користувачів

Для створення чистої бази даних ми будемо використовувати так званий веб-скрепинг це процес збирання даних з сайту шляхом маніпуляції з їх завантаженою сторінкою

Для полегшення процесу розробки на комп'ютері встановлювався інтерфейс Firebase CLI.

Після створення проекту Firebase, ми ініціалізуємо функції Firebase, створюючи базову структуру для функцій.

Залежності, необхідні для веб-скрепинг, такі як Cheerio або Axios, будуть визначені та встановлені в проєкт функцій Firebase.

Оскільки всі такі хмарові функції є по суті інкапсульований nodejs функціями нам потрібно написати функцію для скрапінгу та очищення даних на JavaScript ця функція буде брати дані з сайтів перевіряти їх після чого створювати новий об’єкт у нашій базі даних. Після написання функції її потрібно завантажити як хмарову функцію на firebase також в

тіло виклику функції потрібно прописати як часто ця функція буде запускатися і за яких умов у моїй реалізації функція завантажувала дані з трьох найпопулярніших сайтів притулків та запускається раз на день для оновлення даних у моїй базі даних так ми та отримали очищені дані які можна використовувати в мобільному застосунку.

Ще однією важливою функцією нашого бекенду є авторизація та автентифікація користувача Firebase надає для цього спеціальний сервіс Firebase Authentication яким ми та скористаємося для того, щоб налаштувати автентифікацію користувача нам потрібно спочатку ідентифікувати пристрій з акаунтом і проєктом на Firebase для цього потрібно згенерувати спеціальний захищений криптографією відбиток застосунку це можна зробити через командний рядок gradle після чого ми отримаємо SHA1 відбиток який потрібно додати в налаштуваннях проєкт у відповідь Firebase надішле нам ключ за яким можна буде працювати з автентифікацією.

Після цих дій автентифікацію на бекенд частині можна вважати зробленою.

3.2 Огляд фронтенд частини застосунку

На мою думку, найважливішою частиною на початку розробки мобільних застосунків є розробка правильної архітектури й структури для проекту таким чином наш застосунок буде легко тестувати та розширювати новим функціоналом, Як основну архітектури я обрав Clean Architecture вона описана у другому розділі та власне для реалізації presentation слою було обрано підхід MVVM. Для реалізації внутрішніх баз даних в середині системи андроїд було обрано room, а для ін’єкції залежностей було використано DaggerHilt.Загальна структура проекту зображена на рисунку 3.2.

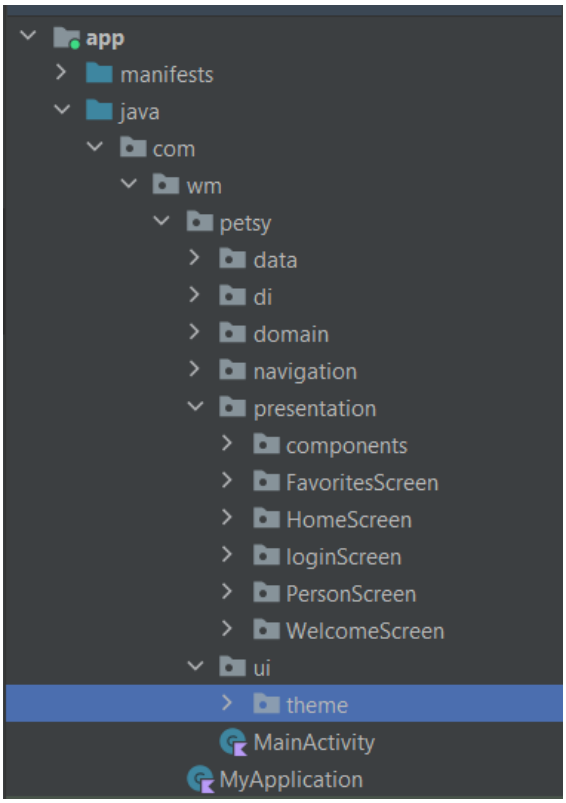

Рисунок.3.2 загальний вигляд структури проекту

Перше що буде бачити користувач при першому використанні застосунку це так звані “OnBoarding Screens” це по суті декілька екранів на яких користувачу буде розказано як користуватися застосунком.Як тільки користувач закінчить “OnBoarding Screens” він більше ніколи їх не бачить тому, що вже знає як користуватися застосунком.

На рисунку 3.3 зображено перший з таких екранів який розказує користувачу про відповідальність за тварину.Наступний екран з процесу ознайомлення зображено на рисунку 3.4 на ньому розказано про спосіб життя з твариною.Останній з екранів зображений на рисунку 3.5 знайомить користувача з процесом адопції.

Ці екрани мають наступний вигляд:

Рисунок.3.3 Перший екран процесу ознайомлення

Рисунок.3.4 Другий екран процесу ознайомлення

Рисунок.3.5 Третій екран процесу ознайомлення

Наступний екран являє собою логін для користувача де він може обрати свій обліковий запис гугл для авторизації екран зображено на рисунку 3.6.

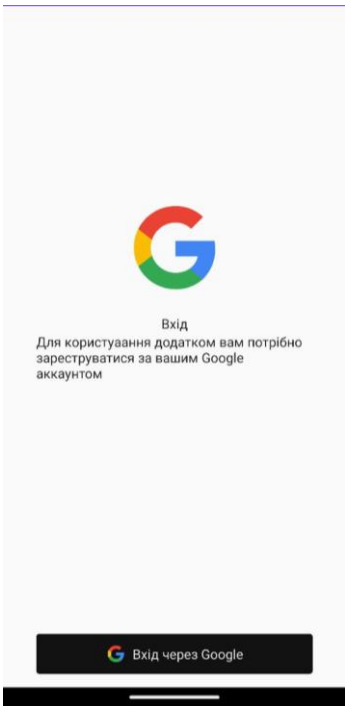

Рисунок. 3.6 Екран авторизації користувача

Після натискання на кнопку у користувача відкриється спеціальне вікно для вибору акаунту за яким він би хотів авторизуватися. Якщо користувач має лише один обліковий запис гугл підключений до пристрою його буде авторизовано автоматично.Вікно вибору облікового запису зображено на рисунку 3.7.

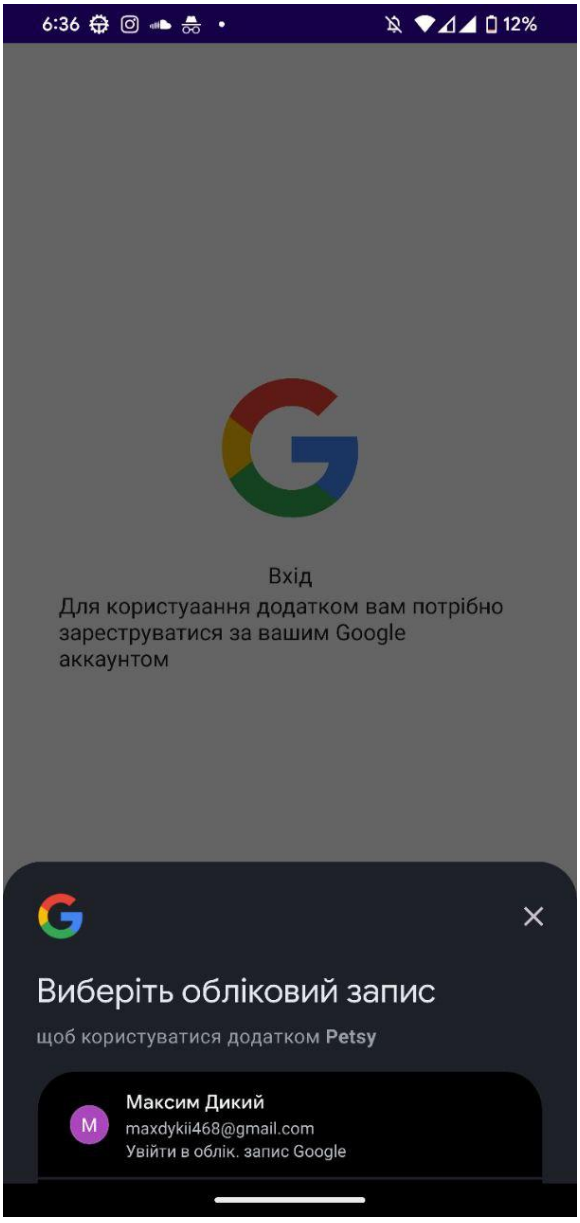

Рисунок. 3.7 Екран вибору акаунту авторизації користувача.

Наступний екран це основний екран на якому і розміщені тварини які можуть бути взяті користувачем зображено на рисунку 3.8.

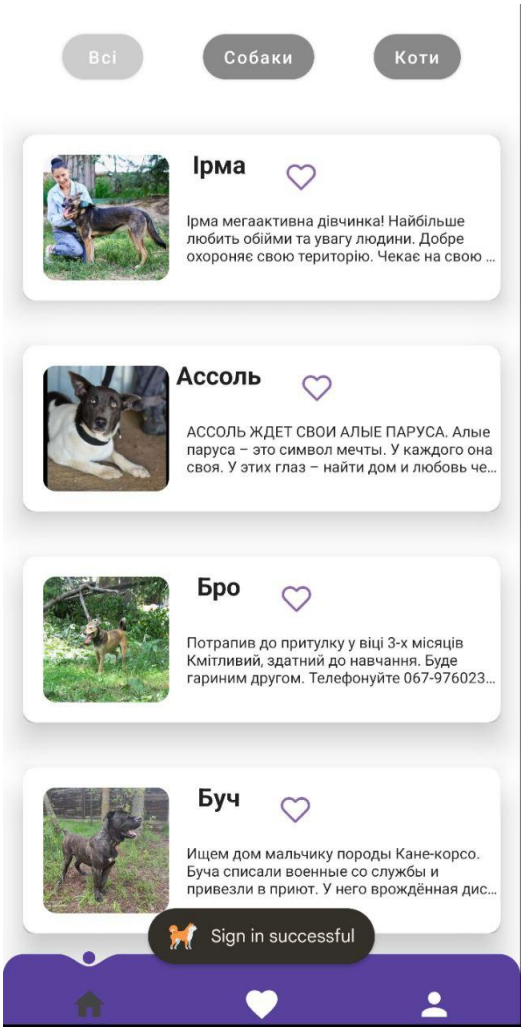

Рисунок.3.8Основний екран застосунку.

З верху екрана видно категорії за якими користувач може обирати яку тварину він хоче взяти в залежності від обраної категорії список зміниться.

На кожній картці тварини у користувача є можливість натиснути на картку і перейти до екрана детальної інформації та додати тварини до категорії улюблених на яку можна перейти скориставшись нижньою панеллю навігації.

Як видно на рисунку 3.8 далі у користувача є декілька нижніх вкладок для взаємодії.

Екран детальної інформації зображено на рисунку 3.9.

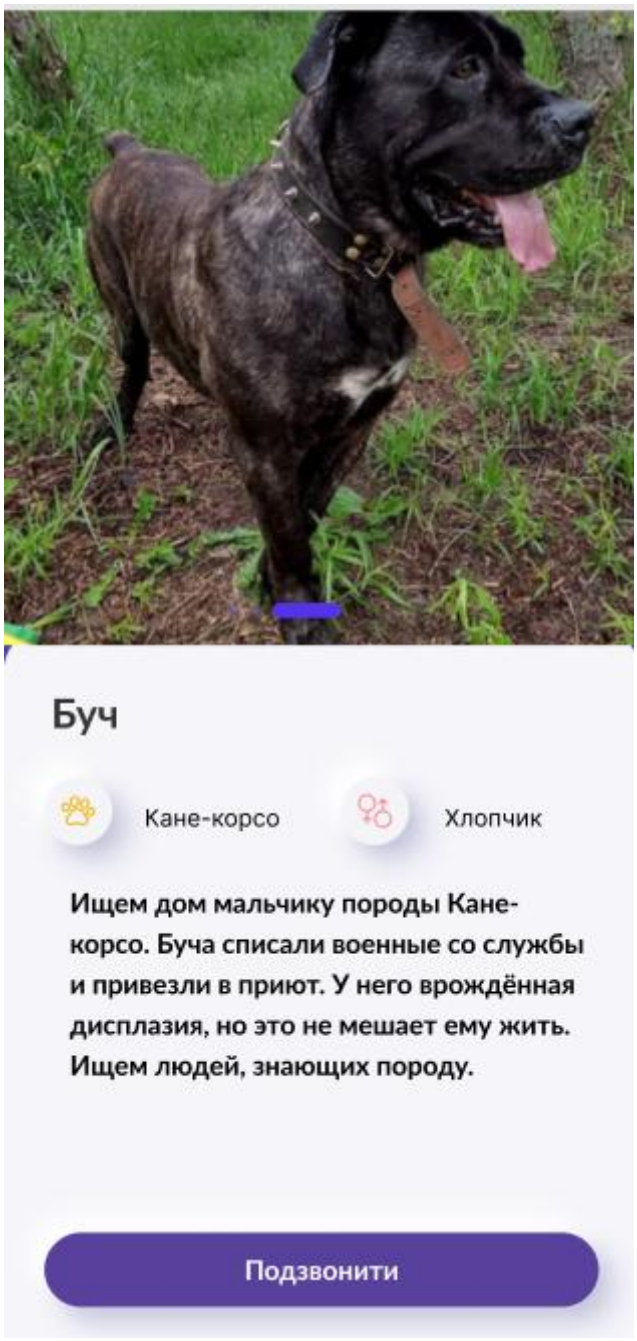

Рисунок. 3.9 Екран детальної інформації про тварину.

Основна це власне список тварин наступна вкладка це улюблені тварини туди користувач може додати тварин які йому сподобалися та розмірковувати над її адопцією (зображено на рисунку 3.10).

Рисунок. 3.10 Екран улюблені тварини.

Наступний екран це управління обліковим записом користувача на ньому він може видалити всю інформацію про свій акаунт або вийти з застосунку . Екран зображено на малюнку 3.11.

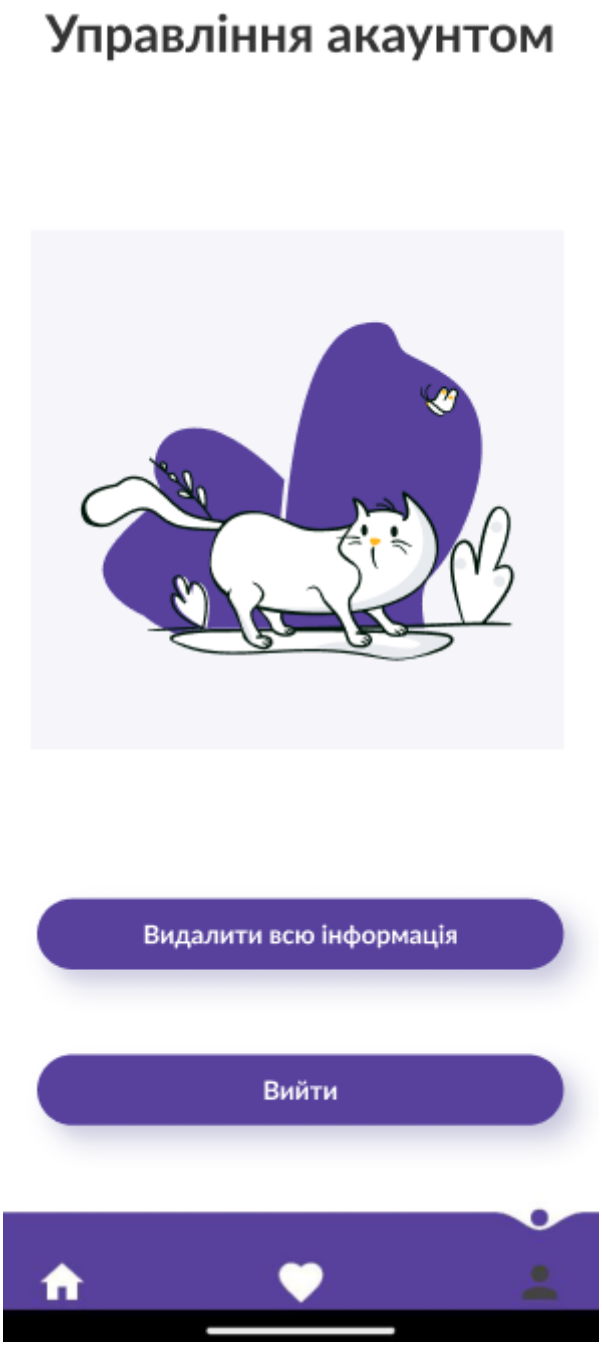

Рисунок. 3.11 Екран управління обліковим записом.

Як результат практичної частини кваліфікаційної роботи ми отримали сучасний мобільний застосунок з приємним дизайном та актуальними даними в ньому які оновлюються кожні 24 години.

ВИСНОВОК

Під час виконання кваліфікаційної роботи бакалавра ми дослідили основні загальнотеоретичні засади та поняття пошуку домашніх тварин, а також вивчили специфіку інформаційних систем, що використовуються для цього. Ми також проаналізували нормативно-правову базу, що регулює пошук домашніх тварин в Україні та світі.

У ході дослідження ми вивчили різноманітні програмні технічні рішення для побудови інформаційних систем, які використовуються для пошуку домашніх тварин. Ми провели глибокий аналіз цих рішень, зосередившись на їхньому змісті та потенційних перевагах для користувачів.

Крім того, ми дослідили мобільні системи та їхню історію і причини створення. Ми розглянули основні поняття, що пов'язані з мобільними системами, і вивчили сучасні підходи до реалізації мобільних застосунків під платформу Android.

На основі набутих знань та досліджень ми спроектували, розробили та впровадили зручний та сучасний мобільний застосунок для пошуку та адопції домашніх тварин з дотриманням усіх необхідних інженерних вимог. Наш застосунок дозволяє користувачам швидко та ефективно знаходити та приймати усвідомлене рішення щодо придбання домашнього улюбленця.

Ця робота покриває широкий спектр тем, пов'язаних з пошуком домашніх тварин та розробкою мобільних застосунків, і ми надіємося, що вона буде корисною для подальших досліджень у цій галузі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ЗАКОН УКРАЇНИ №3447-IV.: веб-сайт. URL: https://www.president.gov.ua/documents/3447-iv-3976(Дата Звrення 10.11.2022).

2. Порядок утримання домашніх тварин : веб-сайт. URL: https://shorturl.at/qLQR6(Дата Звернення 10.11.2022).

3. Європейська конвенція про захист домашніх тварин. : веб-сайт. URL: https://zakon.rada.gov.ua/laws/show/994_a15#Text (Дата Звернення 10.11.2022).

4. Про охорону навколишнього природного середовища. : веб-сайт. URL: https://zakon.rada.gov.ua/laws/show/1264-12#Text (Дата Звернення 10.11.2022).

5. Про захист населення від інфекційних хвороб. : веб-сайт. URL: https://zakon.rada.gov.ua/laws/show/1645-14 (Дата Звернення 10.11.2022).

6. International fund for animal welfare IFAW. : веб-сайт. URL : https://www.ifaw.org/international (Дата Звернення 10.11.2022).

7. World animal protection international. : веб-сайт. URL: https://www.worldanimalprotection.org/ (Дата Звернення 10.11.2022).

8. Українське об'єднання захисників тварин UAAA. : веб-сайт. URL: http://uaaa.org.ua/uk/ (Дата Звернення 10.11.2022).

9. Звіт за 2022 рік UAnimals : веб-сайт. URL: https://uanimals.org/report/zvit-2022/ (Дата Звернення 10.11.2022).

10. Happy Paw. Благодійний фонд Happy Paw .: веб-сайт. URL: https://happypaw.ua/ua/guardianship (Дата Звrення 10.11.2022).

11. UPAW організація для добробуту тварин.: веб-сайт. URL: https://upaw.org/ua/ (Дата Звернення 10.11.2022).

12. Muschko B. Gradle in Action 1st Edition / Benjamin Muschko., 2012 C.12-28.

13. SQLite посібник : веб-сайт. URL:
http://www.w3big.com/ru/sqlite/default.html (Дата Звернення 10.11.2022).
14. Посібник по SQLite : веб-сайт. URL:
https://shorturl.at/UY239(Дата Звернення 10.11.2022).
15. SQLite Documentation. : веб-сайт. URL:
https://www.sqlite.org/lang.html (Дата Звернення 10.11.2022).
16. Kreibich J. A. Using SQLite / Jay A. Kreibich., 2010.,C.76
17.Android developers blog. : веб-сайт. URL: https://android-developers.googleblog.com/ (Дата Звернення 10.11.2022).
18. Basics linux/unix commands : веб-сайт. URL:
https://shorturl.at/htu78(Дата Звернення 10.11.2022).
19. Horton J. Android Programming for Beginners: Build in-depth, full-featured Android apps starting from zero programming experience, 3rd Edition / JohnHorton., 2021. – 742 с.
20. Kunal D. A. Gradle Essentials / Dabir Abhinandan Kunal., 2015. – 176 с.
21. Berglund T. Building and Testing with Gradle: Understanding Next-Generation Builds / T. Berglund, M. McCullough., 2011. – 116 с.
22. Pelgrims K. Gradle for Android / Kevin Pelgrims., 2015. – 172 с.
23. Комлев Н. Объектно-ориентированное программирование / Николай Комлев., 2020 - 29с.
24.Basic unix commands. : веб-сайт. URL:
https://shorturl.at/dqPW4(Дата Звернення 10.11.2022).
25. History of Android Systems : веб-сайт. URL:
https://www.javatpoint.com/android-history(ДатаЗвернення10.112023)
26. Firebase cloud functions : веб-сайт. URL:
https://shorturl.at/hmxyM(Дата Звернення 10.11.2022).
26. History of Linux.: веб-сайт. URL:
https://en.wikipedia.org/wiki/History_of_Linux (Дата Звернення 10.11.2022).
27. Professional Android: Reto Meier, Ian Lake – 2018 - 89-93с.

28. Unix filesystemedia.org : веб-сайт. URL:
https://en.wikipedia.org/wiki/Unix_filesystem (Дата Звернення 10.11.2022).
29. Dundar O. Expert Android Studio / O. Dundar, M. Yener., 2016.
30. AMC College. Android Studio: Apps Development / AMC Coll - 2012-12c.
31.History of Android . : веб-сайт. URL:
https://shorturl.at/aeiAF(Дата Звернення 10.11.2022).
32.Getting started with firebase cloud.: веб-сайт. URL:
https://shorturl.at/jvx17(Дата Звернення 10.11.2022).
33.Introduction to kotlin language android. : веб-сайт. URL:
https://shorturl.at/qyO46(Дата Звернення 10.11.2022).
34.Introduction to kotlin general. веб-сайт. URL:
https://shorturl.at/isOW1(Дата Звернення 10.11.2022).
35 Introduction to Kotlin basics.:веб-сайт. URL:
https://shorturl.at/gtxL3(Дата Звернення 10.11.2022).
36.Kotlin | Посібник .: веб-сайт. URL:
https://metanit.com/kotlin/tutorial/ (Дата Звернення 10.11.2022).
37.Kotlin introduction.: веб-сайт. URL:
https://shorturl.at/oNST0(Дата Звернення 10.11.2022).
38.Kotlin language specification.: веб-сайт. URL:
https://kotlinlang.org/spec/introduction.html (Дата Звернення 10.11.2022).
39.Linux/Unix tutorial.: веб-сайт. URL:
https://www.javatpoint.com/linux-tutorial (Дата Звернення 10.11.2022).
40.The (updated) history of Android. : веб-сайт. URL:
https://shorturl.at/bdFZ2(Дата Звернення 10.11.2022).
41.Unix / linux - file system basics.: веб-сайт. URL:
https://shorturl.at/nJKM4(Дата Звернення 10.11.2022).
42.UNIX operating system.: веб-сайт. URL:
https://shorturl.at/dvBDK(Дата Звернення 10.11.2022).

43.Introduction to linux OS.: веб-сайт. URL: https://www.guru99.com/introduction-linux.html (Дата Звернення 10.11.2022).

44.Linux operating system.: веб-сайт. URL: https://www.javatpoint.com/what-is-linux (Дата Звернення 10.11.2022).

45. Androidpolice.com.: веб-сайт. URL: https://shorturl.at/aeHP6 (Дата Звернення 10.11.2022).

46. Operating System Market Share Ukraine Unter.: веб-сайт. URL: https://shorturl.at/oDESZ (Дата Звернення 10.11.2022).

47. Mobile Operating System Market Share Ukraine: веб-сайт. URL: https://shorturl.at/tG267(Дата Звернення 10.11.2022).

48. Android Developers .android.com: веб-сайт. URL: https://developer.android.com (Дата Звернення 10.11.2022).

49. Clean Coder ncoder.com: веб-сайт. URL: https://shorturl.at/wAQTY(Дата Звернення 10.11.2022).

50. Cloud Firestore Google.com.: веб-сайт. URL: https://shorturl.at/tuITW(Дата Звернення 10.11.2022).

51. Kathuria A. Challenges in Android Application Development: A Case Study / A. Kathuria, A. Gupta. // International Journal of Computer Science andMobile Computing. – 2015. – С. 294–299.

52. Head First Object-Oriented Analysis Design / Brett D. McLaughlin, Gary Pollice & David West., – 2006

53. Professional Android Application Development/Reto Meier,-2012.-253с.

54.Androids: The Team That Built the Android Operating System Paperback – August 12, 2021.,-С.259-270.

55. Nudelman G. Android Design Patterns / Greg Nudelman., 2013.

ДОДАТКИ

Додаток А

```
class GoogleAuthUiClient (
    private val context: Context,
    private val oneTapClient: SignInClient
    ) {
        private val auth = Firebase.auth

        suspend fun signIn(): IntentSender? {
            val result = try {
                oneTapClient.beginSignIn(
                    buildSignInRequest()
                ).await()
            } catch(e: Exception) {
                e.printStackTrace()
                if(e is CancellationException) throw e
                null
            }
            return result?.pendingIntent?.intentSender
        }

        suspend fun signInWithIntent(intent: Intent): SignInResult {
            val credential = oneTapClient.getSignInCredentialFromIntent(intent)
            val googleIdToken = credential.googleIdToken
            val googleCredentials = GoogleAuthProvider.getCredential(googleIdToken, null)
            return try {
                val user = auth.signInWithCredential(googleCredentials).await().user
                SignInResult(
                    data = user?.run {
                        UserData(
                            userId = uid,
                            username = displayName,
                            profilePictureUrl = photoUrl?.toString()
                        )
                    },
                    errorMessage = null
                )
            } catch(e: Exception) {
                e.printStackTrace()
                if(e is CancellationException) throw e
                SignInResult(
                    data = null,
                    errorMessage = e.message
                )
            }
        }

        suspend fun signOut() {
            try {
                oneTapClient.signOut().await()
                auth.signOut()
            } catch(e: Exception) {
                e.printStackTrace()
                if(e is CancellationException) throw e
            }
        }
```

Рисунок А1 – Клас відповідальний за автентифікацію користувача.

Додаток Б

```
@ExperimentalAnimationApi
@ExperimentalPagerApi
@Composable
fun SetupNavGraph(
    navController: NavHostController,
    startDestination: String
) {
    NavHost(
        navController = navController,
        startDestination =startDestination
    ) {
        composable(route = Welcome.route) {
            WelcomeScreen(navController = navController)

        }
        composable(route = Home.route){
            HomeScreen(navController = navController)
        }
        composable(route = Login.route){
            SigninScreen(navController =navController )
        }
        composable(route =Favorite.route){
            FavoriteScreen(navController =navController )
        }
        composable(route =Person.route){
            PersonScreen(navController =navController)
        }
        composable(route =OnBoarding.route){
           WelcomeScreen(navController = navController)
        }
        composable(route = Preview.route){
            val petJson = it.arguments?.getString("pet")
            val moshi = Moshi.Builder().build()
            val jsonAdapter = moshi.adapter(PassString::class.java).lenient()
            val petObject = jsonAdapter.fromJson(petJson)
            if (petObject != null) {
                AnimalPreview(pet = petObject.Animalid)
            }
        }

        }
    }
```

Рисунок 2Б – Основний навігаційний граф застосунку.