ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра інженерії програмного забезпечення

ПОЯСНЮВАЛЬНА ЗАПИСКА

до бакалаврської кваліфікаційної роботи

на ступінь вищої освіти бакалавр

на тему: «**РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ ДЛЯ ВЛАСНИКІВ ТВАРИН НА ОСНОВІ NODE.JS ТА REACT NATIVE**»

Виконала: студентка 5 курсу, групи ППЗ– 51
спеціальності
121 Інженерія програмного забезпечення
(шифр і назва спеціальності)
Шевцова Т.І.
(прізвище та ініціали)
Керівник Жебка В.В.
(прізвище та ініціали)
Рецензент_____
(прізвище та ініціали)
Нормоконтроль _____
(прізвище та ініціали)

Київ – 2022

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти - «Бакалавр»

Спеціальність - 121 Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного
забезпечення
_ _____О.В. Негоденко
«____»_____2022 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Шевцова Тетяна Ігорівна
(прізвище, ім'я, по батькові)

1. Тема роботи: «Розробка мобільного додатку для власників тварин на основі Node.js та React Native»

Керівник роботи Жебка Вікторія Вікторівна, доцент кафедри
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від "18.02.2022 №___"

2. Строк подання студентом роботи 03.06.2022

3. Вихідні дані до роботи:
   - 3.1 Visual Studio Code.
   - 3.2 Офіційна документація React Native.
   - 3.3 Figma.
   - 3.4 MongoDB

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):

4.1 Огляд існуючих мобільних додатків для власників домашніх тварин

4.2. Визначення набору необхідних інструментів для написання мобільного додатку.

4.3 Візуалізація та створення макету мобільного додатку.

4.4 Висновки

5. Перелік графічного матеріалу

5.1 Титульний слайд
5.2 Мета, об’єкт та предмет дослідження
5.3 Аналоги
5.4 Порівняння з аналогами
5.5 Технічні завдання
5.6 Програмні засоби реалізації
5.7 Апробація результатів дослідження
5.8 Методи та класи програми
5.9 Результати створення мобільного додатку
5.10 Висновки
5.11 Кінцевий слайд

6. Дата видачі завдання 11.04.2022

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Вибір теми бакалаврської роботи	16.04.2022	Виконано
2	Підбір науково-технічної літератури	17.04.2022	Виконано
3	Дослідження аналогів та актуальності додатку	18.04.2022	Виконано
4	Аналіз та вибір інструментів для розробки додатку	19.04.2022	Виконано
5	Проектування та реалізація	20.04.2022-29.04.2022	Виконано
6	Висновки, оформлення роботи	02.05.2022	Виконано
7	Розробка демонстраційних матеріалів	03.05.2022	Виконано

8	Попередній захист роботи	26.05.2022	
8	Здача роботи	06.06.2022	

Студент _____ Шевцова Т.І.
(підпис) (прізвище та ініціали)

Керівник роботи_____ Жебка В.В.

ЗМІСТ

РЕФЕРАТ

Текстова частина бакалаврської роботи 69 с., 57 рис., 46 джерел.

Актуальність розробки – на сьогоднішній день домашні улюбленці є майже в кожній родині. Проте кількість безпритульних тварин також велика, оскільки деякі господарі відмовляються від підопічних і здають в притулок або ж залишають на вулиці. Через це з’являються все більше волонтерських організацій, таким чи іншим чином пов ‘язаними одна з одною, оскільки спільна допомога є ефективнішою, і власноруч допомагають тим, хто опинився на вулиці. Тож це унікальний мобільний додаток, що буде корисним як для власників домашніх тварин, так і для волонтерів, аби створити умови задля ефективної допомоги тваринам.

Об'єкт дослідження – процес догляду за тваринами (як домашніми, так і безпритульними).

Предмет дослідження – програмне забезпечення, що сприятиме більш ефективній допомозі тваринам.

Мета дослідження – покращення процесу допомоги тваринам в режимі реального часу з допомогою застосування мобільного додатку, написаного на React Native та Node.js.

У даній дипломній роботі було проаналізовано наявні аналоги, їх переваги та недоліки, обрано програмні інструменти для розробки додатка.

Описано програмні засоби, які були використані у ході роботи.

Додаток реалізований за допомогою мови програмування JavaScript, зокрема, фреймворку React Native для кросплатформенної розробки та Node.js для розробки серверної частини.

Галузь застосування – додатком будуть користуватися люди, які мають домашню тварину, щоб записувати певні дані про неї, додавати нагадування щодо важливих справ та спілкуватися з іншими власниками тварин, а також волонтери, що прагнуть допомагати тваринам.

Ключові слова – Волонтери, власники тварин, React Native, Node.js, MongoDB, компонент, стор, iOS, Redux.

ВСТУП

На сьогоднішній день існує дуже багато організацій, що активно пропагують гуманне ставлення до тварин, особливо до безпритульних. Групи людей-однодумців об'єдналися у всьому світі, щоб стати на захист наших молодших братів та сестричок. Вони регулярно проводять акції, вимагаючи прийняти законопроєкти, що посилюють відповідальність за жорстоке поводження по відношенню до тварин, допомагають притулкам, заповідникам, фізично та фінансово, або ж самі створюють власні притулки та рятують життя десятки тисяч беззахисних.

Робота цих надзвичайних людей тільки декілька років тому почала бути помітною для великої кількості людей. Відносно недавно, завдяки соцмережам, волонтери почали освітлювати свою діяльність широкому колу аудиторії, показувати повсякденне життя тваринок у вольєрах, а також надсилати списки товарів, продуктів, які необхідні хворим, інвалідам, та й звичайним хвостикам. Завдяки цьому я переосмислила повністю те, наскільки проблема безпритульних тварин, а також чорного розведення та знущання над ними є великою та болісною. Проте ще гірше усвідомлювати, яка велика кількість тих, кого ще не знайшли і кому так потрібна допомога.

Також є організації та команди по порятунку тварин з надзвичайних ситуацій. Волонтери відгукуються на запити щодо тварин, що потрапили у небезпеку, або через власну неуважність, або через безвідповідальність своїх господарів, виїжджають на виклики, незважаючи на те, ранок це, чи глибока ніч. В наші дні особливо важливо мати ефективний та швидкий спосіб комунікації. Наразі більшість таких випадків фіксують на мобільні пристрої та викладають у соцмережах, відмічають служби порятунку тварин, а ті вже відповідно реагуютьПроте, завжди тільки сподіватися на допомогу волонтерів, до речі, людей, які в більшості випадків, не отримують кошти за свою працю, також не треба. Так, вони завжди готові допомогти інформаційно, якщо ситуація критична – і фізично,

але нам не потрібно перекидати проблеми на інших, а намагатися вирішати їх самостійно, або, хоча б також брати в цьому участь.

Саме це і дало мотивацію розробити додаток, який, перш за все, буде максимально зручним та простим у використанні, при цьому оснащений необхідним функціоналом для ефективної допомоги безпритульним тваринам, волонтерам, притулкам.

Мета дослідження – покращення процесу допомоги тваринам в режимі реального часу з допомогою застосування мобільного додатку, написаного на React Native та Node.js.

Завдання дослідження:

1. Проаналізувати існуючі аналоги
2. Зробити аналіз технічних засобів та інструментів для розробки, обрати, що найкраще підійде до даного додатку
3. Підготовити структуру файлів проекта, встановити всі необхідні залежності
4. Розробити клієнтську частину проекту
5. Розробити серверну частину
6. Провести тестування застосунку, перевірити, чи відповідає він початковим умовам

Об'єкт дослідження – процес догляду за тваринами (як домашніми, так і безпритульними).

Предмет дослідження – програмне забезпечення, що сприятиме більш ефективній допомозі тваринам.

Сфери застосування – в повсякденному житті для введення щоденнику домашньої тварини та для повідомлення волонтерам про тварин, що потребують допомогу.

1. АНАЛІЗ І ВИДІЛЕННЯ ПЕРЕВАГИ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальна статистика безпритульних та зниклих тварин, їх причини

Дуже часто, прогулюючись містом, ми бачимо бездомних тварин, в більш віддалених місцях навіть цілі зграї. В деяких є нашийники, чи кліпси, але вони не мають господарів. Вони є безпритульними. Ці тварини виживають в суворих умовах на вулицях, кожного дня наражаються на небезпеку або від інших тварин, або від людей. Особливо від людей, від підлітків, оскільки ті знають, що за це їм не буде покарання. До речі, варто зазначити, що насильство над тваринами породжує насильство над людьми. Ще з психології відомо, що діти, які схильні до жорстокого поводження з беззахисними тваринами, мають проблеми в сім'ї та в стосунках з близькими.

Спробуємо трішки поглибитись в історію та розібратись, чому ж безпритульні тварини з'явилися на вулицях. Як відомо, першою твариною, яку людині вдалося приручити, є вовк. Він став супутником людини, прибившись до неї в кам'яну добу, приблизно 15 тисяч років тому. Одомашненим підвидом вовка є собака. А ось котиків приручили десь на 5 тисяч років пізніше. Роль цих тварин в тодішньому суспільстві була різною. Наприклад, собаки часто допомагали людям у мисливстві, їх брали на полювання та ловили здобич. Також роль собаки була й охорона помешкання. Коти допомагали у боротьбі з гризунами. Цих тварин ще утримували і для розваг та спілкування. Вони ставали справжніми членами сім'ї. Проте саме це і вплинуло на подальшу долю цих тварин. Через одомашнення вони втратили більшу частину рис, які були притаманні їх диким предкам. Вони стали дуже залежними від людини. Людина надає все те основне, чого вони потребують – їжу, воду, тепле місце, увагу і любов. Тварини стали вірними друзями та компаньйонами людини, а люди стали відповідальними за їх життя.

Дуже велика кількість котів та собак опинилась на вулиці. Саме через безвідповідальність людей. Через тих людей, котрі вирішили переїхати до іншого

міста чи країни, та кинути напризволяще своїх підопічних, або ж в “кращому” випадку, передати до притулків, які і так завжди переповнені. Через тих, кому набридло прибирати, вигулювати, виховувати, приділяти свій час. Є ті, хто передумав, та коли цуцик чи кошеня виростає і потребує певного виховання, від них також відмовляються. Левова частка тварин, що були викинуті з чорного розведення, оскільки вже не здатні народжувати. Ось чому в притулках можна бачити в тому числі і породистих собак чи котів. Їх або врятовують з розведення, або власники просто не змогли впоратися з потребами тої чи іншої породи. Опинившись на вулиці, тварини не перестають розмножуватись, їх кількість значно збільшується. Через це з’являються цілі зграї, небезпечні для людей, тварини стають безпритульними спадково.

Ще раніше люди вживали різних заходів по контролю росту безпритульних тварин. Їх або топили ще змалечку, або підсипали в їжу отруйну речовину. Навіть важко уявити інші способи. Замість того, щоб стерилізувати, їх також піддавали евтаназії (міри зі швидкого переривання біологічних процесів в організмі). Проте тут варто розуміти, що проблему це не вирішить, адже знищення однієї тварини не зупинить розмноження десятків інших, що залишились на вулиці. Також населення не було достатньо проінформовано цією проблемою, тому кількість бездомних тварин зростала.

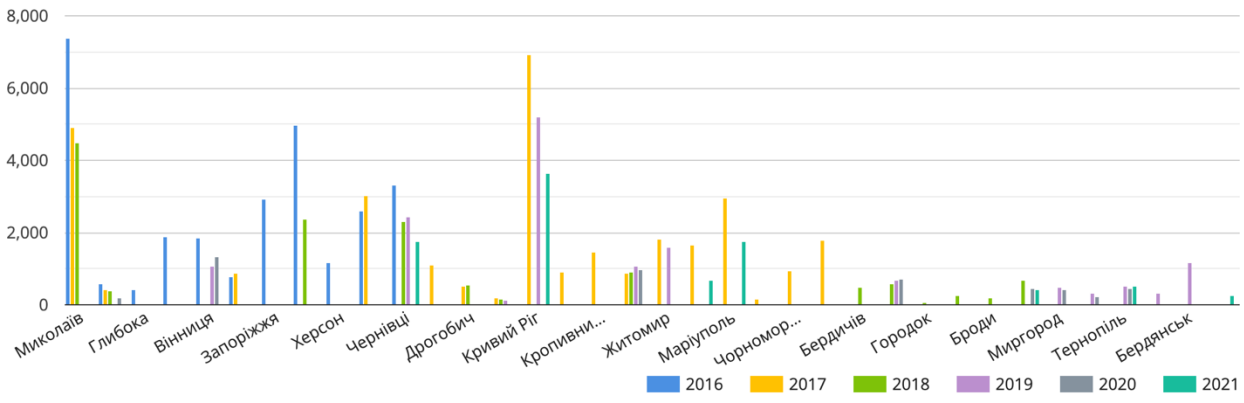


Рисунок 1.1 (Статистика, щодо тварин на вулицях по всій Україні)

На сьогоднішній день платформа Animal-id збирає та поширює актуальні дані, щодо тих тварин, які опинилися на вулицях. Зокрема, станом на 2022 рік

максимальна кількість тварин спостерігається в містах: Миколаїв (4503), Кривий Ріг (3645), Київ (3034), Запоріжжя (2918), Дніпро (2358), Одеса (1882), Ізмаїл (1800), Маріуполь (1754). Кількість цих тварин також підраховується й волонтерами в рамках проектів. Але точне число безпритульних тварин на території всієї України назвати важко, адже воно постійно змінюється.

Загалом такі тварини пристосовуються до нового способу життя, вони обирають собі місце перебування, інколи це такі двори чи території, де їх підгодовують люди та використовують для охорони. В такому випадку, тварини не є небезпечними для суспільства, бо вони все ж контактують с людьми, отже їм довіряють. Проте, також є випадки, коли у тварин є моральні травми , вони пережили стрес через негативний досвід з людиною, тому вони повністю ізолюються та агресують, коли їм намагаються допомогти, бо хочуть захистити себе. Особливо небезпечно, якщо такі тварини об'єднуються у великі зграї.

Як кажуть: "Велич і моральний прогрес нації можна виміряти тим, як ця нація ставиться до тварин.". Отже, є декілька основних заходів, щоб зменшити кількість безпритульних тварин:

1. Отримання ліцензії на їх розведення

Люди, які розводять тварин для продажу, мають отримати на це ліцензію, як, наприклад, це заведено у Великобританії. Також вони зобов'язані показати інспекторам приміщення, в якому утримуються тварини, ветеринарам, аби впевнитися, що тварини здорові та отримують достатню кількість їжі та води.

Проте, зараз дуже популярним є рух Adopt, Not Shop! (Не купуй – прихисти!). Ті, хто бажає породисту собаку чи кота, можуть знайти їх і в притулках. Тому не варто підтримувати чорне розведення.

2. Сплата податку за утримання тварин

Власники тварин повинні сплачувати податки, приклад – Німеччина. Кошти спрямовуються на утримання та допомогу притулкам. А ті, хто саме прихистив тварину, може не сплачувати податок, адже таким чином кількість безпритульних зменшується.

3. Реєстрація тварин

Дуже важливо у випадку, коли тварина загубилась, або ж її просто залишили. Таких тварин , за наявності мікрочіпу, жетону чи кліпси, легко ідентифікувати. В базі даних зберігатимуться відомості про власників, також вік, стать, кличка знайденої тварини, особливості поведінки чи дані про здоров'я. Також реєстрація допомагає встановити кількість покинутих людиною тварин, та тих, які є безпритульними спадково. Таким чином, власників покинутих тварин можна притягнути до відповідальності та позбавити права мати будь-яку тварину в майбутньому.

4. Стерилізація

Також важливий пункт. Відловлених тварин з вулиць потрібно стерилізувати, задля запобігання їх розмноження та збільшення кількості спадкових безпритульних тварин. Для регулювання чисельності безпритульних тварин, в Україні діє комплексний підхід: відлов стерилізація-повернення (ВСП), іноді - відлов-вакцинація-стерилізація-прилаштування або відпуск на місце відлову.

5. Створення організацій та притулків

Навіть якщо тварина стерилізована та зареєстрована, це не виключає факту, що вона може продовжувати залишатись на вулиці. Отже від цього кількість безпритульних також сильно не зміниться. Тому важливим пунктом є створення притулків та центрів адопції. Це також буде заохочувати людей адоптувати тварин, а не купувати у розплідниках за великі гроші.

6. Заборона евтаназії

Метою евтаназії є припинення страждань тих, хто невиліковно хворий і гуманніше було б умертвіння, ніж життя в муках. Це не має жодного відношення до абсолютно здорових тварин, більш того, це не вирішує проблему великої кількості бездомних і не регулює їх чисельність. Зокрема, в Україні вже прийняли закон № 2351, що забороняє жорстоке поводження з тваринами, та їх винищування.

7. Пропагування гуманного ставлення до тварин

Останній, але не менш важливий пункт. Саме завдяки гуманному, відповідальному та свідомому ставленню по відношенню до беззахисних тварин, ми можемо вирішити ці проблеми. Отже, ми маємо розповсюджувати інформацію всіма

доступними засобами, заохочувати інших ставати більш гуманними та відповідальними до тварин та висловлюватись за тих, хто не має цієї можливості.

1.2 Основна мета додатку

Допомога безпритульним тваринам завжди потребує комунікації та розголошення. Бо дуже часто, коли людина бачить таку тварину на вулиці, в неї немає можливості прихистити її, навіть на деякий час. Тому на сьогоднішній день таку роль розповсюдження інформації грають соцмережі. Як правило, фотографують саму тварину, надають її місцезнаходження, а вже спеціальні притулки допомагають з поширенням, або ж самі забирають до себе на перетримку.

Метою даного застосунку є створення додатку, який буде надавати можливість ефективно та швидко передавати інформацію про знайдених тварин, їх геолокацію, до певних служб, а також, щоб інші користувачі також могли бачити цю інформацію та допомагати в пошуках домівки, зборах коштів на лікування. Додаток має бути простим та інтуїтивно зрозумілим для користувачів, а також підходити для різних операційних систем.

1.3 Аналіз наявних аналогів

1.3.1 Animal ID

Популярний додаток та універсальне рішення для відповідальних власників тварин.

1 з 4 домашніх тварин губляться хоч раз в житті. Тому основна ідея закладається в тому, щоб допомогти цим тваринам швидше повернутися до своїх господарів. Яким чином? За допомогою ідентифікатора (QR-коду на спеціальному жетоні) тварини та її профілем в додатку, що вноситься до світової бази ідентифікованих тварин.

Можливості додатку:

– створення профілю тварини

– зберігання документів
– функціональний календар, в якому можна відмічати дати вакцинації, візит до грумера чи прийом таблеток від паразитів
– можливість отримання геоданних загубленої тварини

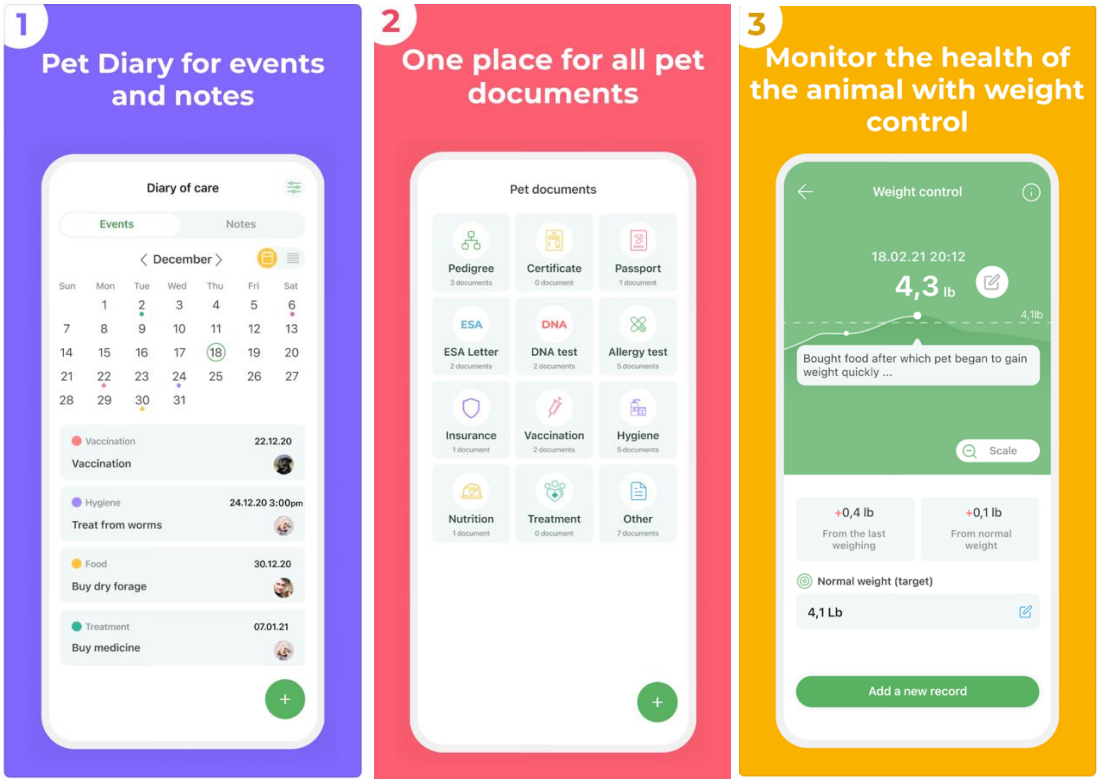


Рисунок 1.2 – Animal ID

У разі, якщо тварина загубилася, вона не зможе назвати іншим свою адресу, чи на який корм в неї алергія. Тому всі данні про неї можна отримати з QR паспорту Animal ID.

Людині, яка знайшла загублену тварину, достатньо відсканувати QR-код на жетоні, щоб дізнатись всю інформацію про тварину та контакти її власника. До речі, QR паспорти працюють по всьому світу.

Коли жетон буде відскановано, власник отримає PUSH-повідомлення та email. Людина, яка відсканувала жетон, зможе вибрати зручний спосіб для зв’язку с власником. Також профіль доступний через пошук по мікрочіпу.

Animal ID задовольняє майже всі потреби для власників тварин, щоб слідкувати за своїм улюбленцем та у випадку, якщо тварина загубилась, мати інформацію про її місцезнаходження за допомогою інших користувачів.

Проте, це не розповсюджується на бездомних тварин, які не мають паспорту чи мікрочіпу. Також неможливість додавати інформацію про їх знаходження на мапу.

Додаток можна скачати за посиланнями:

App Store - https://apps.apple.com/app/animal-id/id1505139116

Google Play - https://play.google.com/store/apps/details?id=com.animalid

1.3.2 Let it Wag

Додаток, орієнтований на допомогу безпритульним тваринам. Він допомагає в режимі реального часу надавати інформацію про знайдену тварину, а вже локальні волонтери, вет. клініки, притулки допоможуть.

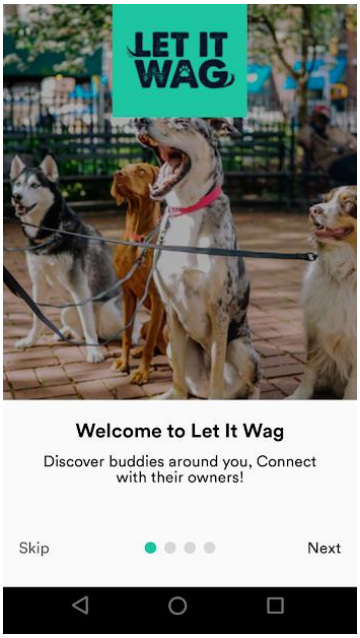


Рисунок 1.2 – Let it Wag

В додатку є декілька функцій:

– сервіс по адопції
– можливість надавати інформацію про тварин, що потребують допомоги

– створення профілю
– сервіс вигулу тварин
– сервіс порятунку
– фінансова підтримка фондів

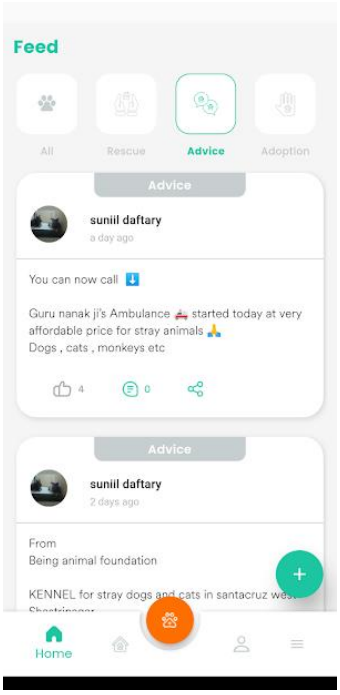


Рисунок 1.3 – Головна сторінка

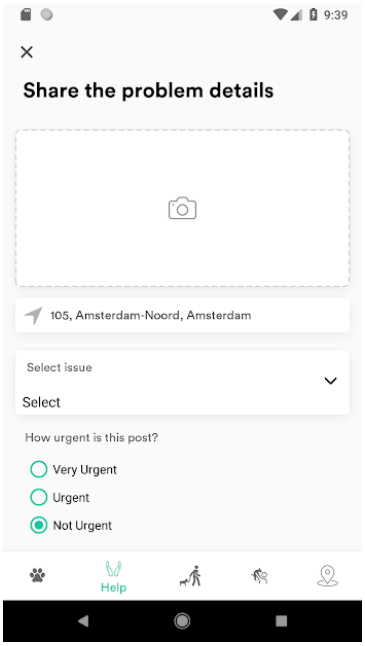


Рисунок 1.4 – Надання інформації про проблему

Відгуки:

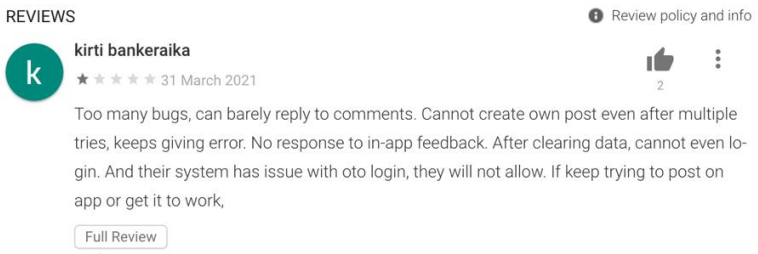


Рисунок 1.5 – Неможливо додати пост

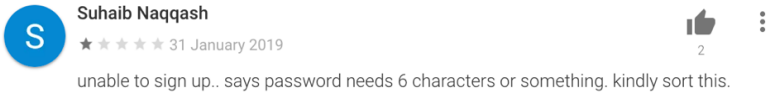


Рисунок 1.6 – Неможливо увійти до додатка

Проаналізувавши відгуки, в додатку наявні баги, що не дають в повній мірі користуватись функціоналом.

Додаток можна скачати за посиланнями:

App Store - https://apps.apple.com/in/app/let-it-wag/id1444641065

Google Play - https://play.google.com/store/apps/details?id=com.letitwag.mobile&hl=en_IN

Таблиця 1.1 – Зведені результати аналізу характеристик додатків для власників тварин

Показник	**Animal ID**	**Let it Wag**	**OnlyPets**
Платформи	Android, iOS, Web	Android, iOS, Web	Android, iOS
Створення профілю	+	+	+
Створення профілю тварини	+	-	+
Оперативна допомога тваринам в	-	+	+

режимі реального часу			
Допомога з прилаштуванням тварин	+	+	+
Наявність календарю для ведення звіту по стану здоров'я тварини	+	-	+
Наявність мапи з геолокацією тварини	+	+	+
Наявність безкоштовної версії	+	+	+
Можливість завантажити документи	+	-	+

1.4 Складання технічного завдання

Щоб реалізувати дану ідею, та зробити її доступною для кожної людини, було вирішено створити саме мобільний застосунок. Адже, людина користується мобільним телефоном більшу частину часу, тож користування додатком буде зручнішим.

По-перше, треба вирішити, які мови програмування будуть використовуватись для розробки клієнтської та серверної частин.

Треба виділити основні способи написання мобільних додатків:

- нативна розробка
- кросплатформенна розробка
- гібридна розробка

Нативні додатки створені для певної операційної системи з використанням специфічних для платформи мов програмування.

Розробка нативних додатків – це процес розробки програм або програмного забезпечення, які повинні працювати на певних пристроях і платформах мобільних додатків, таких як Android та iOS. При розробці додатків розробники покладаються на нативну для операційної системи мову програмування, щоб створити програми, які підходять для однієї конкретної платформи – будь-то настільні комп’ютери, смарт-телевізори, смартфони чи будь-які інші гаджети, які використовуються в цифровому просторі.

Розробка нативних мобільних додатків ідеально підходить, якщо треба забезпечити найкращий користувальницький вигляд з точки зору зовнішнього вигляду та відчуття програми. Завдяки розробці нативних додатків розробники отримують доступ до додавання додаткових можливостей та функцій до програм, оскільки розробка нативних додатків містить можливість використовувати основні елементи апаратного забезпечення смартфона, як-от GPS, датчики наближення, камеру, мікрофон тощо.

Для розробки нативних програм під Android, використовуються:

- Java
- Kotlin

Java — це багатоплатформна, об'єктно-орієнтована та мережево-орієнтована мова програмування.

Kotlin — це мова програмування з відкритим вихідним кодом, яка може працювати на віртуальній машині Java (JVM). Мова може працювати на багатьох платформах.

Для розробки нативних програм під iOS, використовуються:

- Objective-C
- Swift

Objective C є об'єктно-орієнтованою мовою програмування загального призначення.

Swift — це мультипарадигмальна мова програмування загального призначення, яку Apple Inc. розробила для різних операційних систем, таких як iOS, tvOS, macOS, watch OS.

Переваги нативної розробки:

- Висока швидкість

Оскільки нативні мобільні додатки не мають складного коду, як-от гібридні та кросплатформенні програми, вони виявляються порівняно швидшими. Більшість елементів програми швидко відображаються, оскільки вони попередньо завантажуються. Завдяки високій швидкості розробки та економічності, стартапи віддають перевагу нативним додаткам.

- Функціональність офлайн

Однією з головних переваг нативної програми є те, що нативні програми працюють бездоганно навіть за відсутності підключення до Інтернету. Це забезпечує більше зручності для користувачів, оскільки вони отримують доступ до всіх функцій програми в режимі польоту або в автономному режимі. Ця функція підтримки в автономному режимі в нативних додатках є важливою для користувачів додатків, які проживають у районах із низьким рівнем підключення до Інтернету, перебувають у віддалених районах або мають обмежену доступність даних.

- Більш інтуїтивно зрозумілий та інтерактивний

Оскільки нативні програми створені для певної операційної системи, вони дотримуються вказівок, які забезпечують покращений досвід, який ідеально відповідає певній операційній системі в усіх термінах. Крім того, оскільки нативні програми дотримуються інструкцій, це дає змогу користувачам взаємодіяти з програмами за допомогою жестів і дій, з якими вони вже знайомі.

- Мінімізований обсяг помилок

Керування двома різними кодовими базами стає складним завданням у порівнянні з однією кодовою базою, як у розробці нативних програм. Оскільки нативні програми мають одну кодову базу і не покладаються на міжплатформенні інструменти, у них виникає мінімальна кількість помилок.

- Максимальна продуктивність

Розробка нативних програм допомагає розробляти програми, які є оптимізованими для певної платформи, що забезпечує високу продуктивність. Оскільки нативні програми створені для певної платформи, ці програми виявляються порівняно швидкими. Крім того, ці програми компілюються з основними мовами програмування та API, що робить нативні програми більш ефективними.

- Посилена безпека

Розробка нативних додатків покладається на різні браузери та технології, такі як HTML5, JavaScript і CSS, і забезпечує повний захист даних з боку користувача.

Приклади нативних програм:

- Google Maps
- Artsy
- Pinterest
- Spotify

Кросплатформенні фреймворки використовуються для написання спільного та багаторазового коду для створення програм для різних операційних систем. Одноразове написання коду та його повторне використання на кількох платформах допомагає мінімізувати витрати та зусилля на розробку.

Кросплатформенні програми забезпечать безпроблемну реалізацію, надійну функціональність і доступне виробництво. Однак не треба очікувати високої продуктивності та налаштування за допомогою кросплатформенних засобів розробки додатків.

Для розробки кросплатформенних програм використовуються:

- React Native
- Xamarin
- Flutter

React Native — це фреймворк, який використовується для створення кросплатформних мобільних додатків. Це один з найпопулярніших і широко використовуваних кросплатформенних фреймворків для розробки мобільних додатків.

Xamarin був випущений у 2011 році корпорацією програмного забезпечення. Нещодавно, у 2016 році, Microsoft придбала Xamarin. Він дуже популярний у розробці кросплатформних додатків і надає різні інструменти розробки. Це фреймворк для створення кросплатформного мобільного додатка C#.

Flutter — це безкоштовна технологія з відкритим вихідним кодом, розроблена компанією Google у травні 2017 року для створення нативних додатків для Android та iOS за допомогою єдиної кодової бази.

Переваги кросплатформенної розробки:

- Безпроблемна і швидка розробка

Отримання багаторазового коду, підкріпленого покращеною продуктивністю та ефективністю, є справжнім бонусом для розробників і власників бізнесу в довгостроковій перспективі. Саме тут фреймворк розробки кросплатформенних додатків отримує конкурентну перевагу над іншими альтернативами.

- Чудове обслуговування продукту

Розробка кросплатформенних додатків передбачає єдину кодову базу. Оскільки існує лише одна кодова база, тестувати та розгортати виправлення та оновлення стає досить легко, а також випробовувати більшу точність і чудову якість мобільних додатків.

- Знижена вартість

Фреймворк розробки кросплатформенних додатків володіє потенціалом підтримки всіх видів платформ і забезпечує ширше охоплення ринку на глобальному рівні зі збільшенням популярності бренду стартапів, які прагнуть швидкого залучення на ринок. Крім того, кросплатформенні програми забезпечують зниження початкових витрат.

- Можливість повторного використання коду

Коли справа доходить до кросплатформенних програм, розробникам не потрібно щоразу писати унікальний код для кожної операційної системи. Загальну кодову базу можна використовувати для передачі коду на різні платформи.

Приклади кросплатформенних додатків:

- Insightly

- Bloomberg
- Reflectly
- Skype
- Slack

Розробка гібридних додатків — це комбінація нативних і веб-рішень.

Гібридна платформа складається в основному з 2 компонентів - серверний код і нативний переглядач, який завантажується для відображення функцій, і серверної частини.

При розробці гібридних програм код пишеться лише один раз, і той самий код можна використовувати для кількох платформ. Гібридна програма здатна забезпечити продуктивність і користувацький досвід, які дещо наближені до нативних програм, але коли справа доходить до UX та шаблонів навігації, то їх не вистачає у розробці гібридних додатків.

Для розробки гібридних програм використовуються:

- Ionic
- Apache Cordova

Ionic — це фреймворк інтерфейсу користувача з відкритим вихідним кодом для кросплатформенної розробки гібридних додатків.

Переваги фреймворку Ionic:

- Ionic дуже гнучкий, оскільки він побудований на стандартизованих веб-технологіях. Це означає, що легко налаштувати зовнішній вигляд програми за допомогою лише модифікацій HTML і CSS
- Ionic імітує зовнішній вигляд нативної програми завдяки своїй бібліотеці компонентів інтерфейсу користувача. Ці компоненти можна використовувати як готові елементи для побудови графічного інтерфейсу користувача, або їх можна налаштувати. Завдяки веб-компонентам Ionic пришвидшує процес розробки логіки інтерфейсу користувача та збереження оригінального вигляду без додаткових витрат

Apache Cordova є безкоштовною платформою розробки мобільних додатків з відкритим кодом для створення кросплатформенних мобільних додатків за

допомогою CSS3, HTML5 і Javascript. Він також дозволяє створювати програми для Android, iOS та Windows.

Переваги фреймворку Apache Cordova:

- Apache Cordova допомагає розробникам розробляти додаток, використовуючи властивості нативних ресурсів пристрою. Тоді програма зможе отримати доступ до потужних вбудованих функцій, таких як камера телефону, акселерометр, компас, повідомлення, контакти, геолокація, мережа, сповіщення (звук, вібрація) та медіа-сховище
- сумісний на декількох платформах

Переваги гібридної розробки:

- Швидкий час виходу на ринок

Стартапи зазвичай мають намір забезпечити швидшу доставку MVP на ринок. У цьому випадку ідеальним вибором є гібридна структура розробки додатків, оскільки вона дає змогу швидко розробити й запустити додаток на ринок порівняно на ранній стадії.

- Легке обслуговування

Оскільки гібридні додатки базуються виключно на веб-технологіях, підтримувати гібридні додатки легко в порівнянні з нативними та кросплатформенними програмами, які мають складну кодову базу.

- Мінімізація витрат завдяки простоті розробки

Оскільки життєвий цикл розробки гібридних додатків супроводжує уніфіковану розробку, це приносить полегшення для стартапів, які мають обмежений бюджет. Їм не потрібно інвестувати окремо для створення різних версій додатків для кількох платформ. Гібридні фреймворки розробки додатків дозволяють розробникам створювати одну версію і використовувати її для кількох платформ.

- Покращений UI/UX

Гібридні програми об’єднують переваги нативних і веб-програм, забезпечуючи бездоганний і покращений досвід роботи на всіх платформах Android та iOS. Розробка гібридних додатків має легкий інтерфейс програми, який забезпечує надшвидке завантаження вмісту.

Приклади гібридних додатків:

- Instagram
- Evernote
- Gmail
- JustWatch
- NHS
- Airbus Helicopters

Використовувана мова програмування має підходити до наступних вимог додатку:

- створення облікового запису користувача
- можливість додавати документи
- наявність календарю
- карта місцезнаходження тварин, можливість оновлення інформації та додавання нових точок
- наявність Push-повідомлень
- можливість додавати повідомлення про знайдену тварину

Проаналізувавши наявні мови програмування та фреймворки, було вирішено використовувати кросплатформенний підхід, адже кодова база може бути використана під різні платформи, та підтримка й розширення функціоналу буде набагато легшою. Вибрана мова програмування JavaScript з використанням фреймворку React Native.

Для серверної частини використовуватиметься середовище виконання JavaScript - Node.js. Для управління версіями застосунку буде використовуватись git, а для хостингу – github.

Створено UML діаграму зв’язків існуючих елементів додатку, завдяки якому можемо бачити, що від чого буде залежне, та як ті чи інші елементи будуть пов’язані безпосередньо з користувачем.

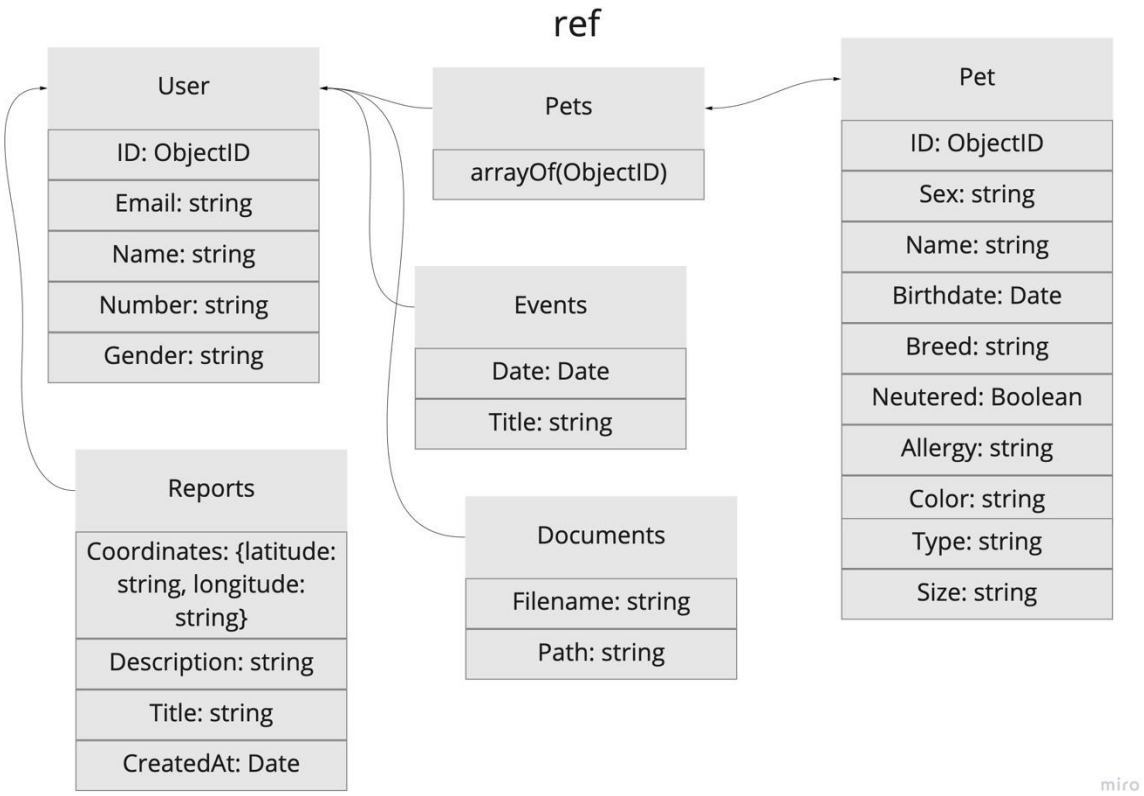


Рисунок 1.7 – UML діаграма бази даних

1.5 Аналіз технологій та засобів розробки

1.5.1 Мова програмування JavaScript

JavaScript - це динамічна мова комп'ютерного програмування. Вона легка і найчастіше використовується для веб розробки, та дозволяє взаємодіяти з користувачем, оброблювати події і створювати динамічні сторінки.

JavaScript дозволяє розробляти не тільки клієнтську частину (наприклад, за допомогою фреймворку Angular або бібліотеки React), а й серверну (Node.js), а також мобільні додатки (з використанням бібліотеки React Native). Отже, головною перевагою мови JavaScript є можливість повторного використання коду для клієнтської та серверної частин.

1.5.2 Середовище виконання Node.js

Node.js — це кросплатформенне середовище виконання з відкритим вихідним кодом для розробки серверних і мережевих додатків. Програми Node.js

написані на JavaScript і можуть запускатися в середовищі виконання Node.js в OS X, Microsoft Windows і Linux.

1.5.3 Figma

Figma — це передовий веб-інструмент для графічного дизайну, який пропонує нові рішення та багато можливостей для створення прототипів UI/UX. Це надійна програма, яка дозволяє створювати унікальні UX-дизайни, які є зручними та професійними.

Використання Figma є дуже простим також і для початківців, та дозволяє створювати дійсно потужні прототипи за допомогою великої кількості наявних інструментів.

1.5.4 Фреймворк React Native

React Native — це фреймворк JavaScript з відкритим вихідним кодом, розроблений для створення додатків на кількох платформах, таких як iOS, Android, а також веб-додатків, використовуючи ту саму кодову базу. Він заснований на React і привносить всю свою славу в розробку мобільних додатків.

1.5.5 Бібліотеки JavaScript

Для встановлення всіх необхідних залежностей буде використовуватись пакетний менеджер **npm** для Node JavaScript платформи. Він встановлює модулі і розумно керує конфліктами залежностей. Його можна налаштувати для підтримки широкого спектру кейсів використання. Найчастіше він використовується для публікації, виявлення, встановлення та розробки програм.

Для відображення помилок буде використана бібліотека react-toastify https://www.npmjs.com/package/react-toastify . Вона дозволяє додавати повідомлення під різні випадки та кастомізувати їх вигляд і інформацію, що буде відображатись.

В додатку буде наявний календар для планування візитів, прийому ліків, нагадування про вигул. Для його імплементації буде використана бібліотека https://github.com/wix/react-native-calendars

Також один з основних пунктів – це мапа, за допомогою якої користувачі зможуть залишати точки зі знайденими тваринами, чи інформацію, де вони бачили загублену тварину крайній раз. Для реалізації мапи буде використано https://github.com/react-native-maps/react-native-maps.

Для зберігання та управління глобальними даними додатку та для запобігання повторюваного коду буде використовуватись бібліотека Redux Toolkit https://redux-toolkit.js.org/. Вона дозволяє писати більш ефективний код, прискорити процес розробки та автоматично застосовувати найкращі рекомендовані методи.

1.5.6 MongoDB

MongoDB — це документно-орієнтована база даних NoSQL, яка використовується для зберігання великих обсягів даних. Замість використання таблиць і рядків, як у традиційних реляційних базах даних, MongoDB використовує колекції та документи. Документи складаються з пар ключ-значення, які є основною одиницею даних у MongoDB. Колекції містять набори документів і функцій, що є еквівалентом таблиць реляційної бази даних. MongoDB — це база даних, яка з’явилася приблизно в середині 2000-х років.

2. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Розробка клієнтської частини

Під час аналізу даної ідеї було вирішено створити додаток для мобільних пристроїв, оскільки це набагато зручніше для користувачів в наш час.

В основному, цільовою аудиторією будуть власники тварин та волонтери, організації, притулки, що допомагають тваринам, оскільки це зможе полегшити комунікацію та пришвидшити процес реагування на певні проблеми.

Головною ідеєю додатку є цифровізація діяльності волонтерів, оперативне поширення інформації, можливість дізнаватись про втрачених чи знайдених тварин за допомогою додатку.

Головні вимоги додатку:

- Профіль користувача
- Профіль тварини
- Наявність календаря
- Додавання нових подій до календаря
- Наявність мапи
- Додавання нових точок з певною інформацією до мапи
- Інформування інших користувачів у випадку зникнення чи знаходження тварини
- Додавання документів

2.1.2 Проектування структури

Для розробки дизайну було обрано Figma, оскільки це дуже потужний інструмент для створення графічних інтерфейсів.

Отже, основними елементами додатку є:

1. Форма для створення профілю (текстові поля для вводу даних та кнопка збереження)
2. Логотип

3. Розділи головної сторінки
4. Кнопка для редагування профілю
5. Кнопка для створення профілю тварини
6. Кнопка для редагування профілю тварини
7. Кнопка для переходу на календар
8. Кнопка для створення нової події
9. Форма для заповнення даних нової події
10. Кнопка для переходу на мапу
11. Кнопка для додавання нової точки на мапу
12. Форма для створення екстреного повідомлення
13. Форма для завантаження документів

Отже, почнемо з розробки дизайну в Figma. Перейдемо до офіційної сторінки та створимо профіль. Після цього створимо новий файл дизайну.

Оскільки це буде мобільний додаток, в опції Frame можемо вибрати прототип, наприклад, моделі iPhone.

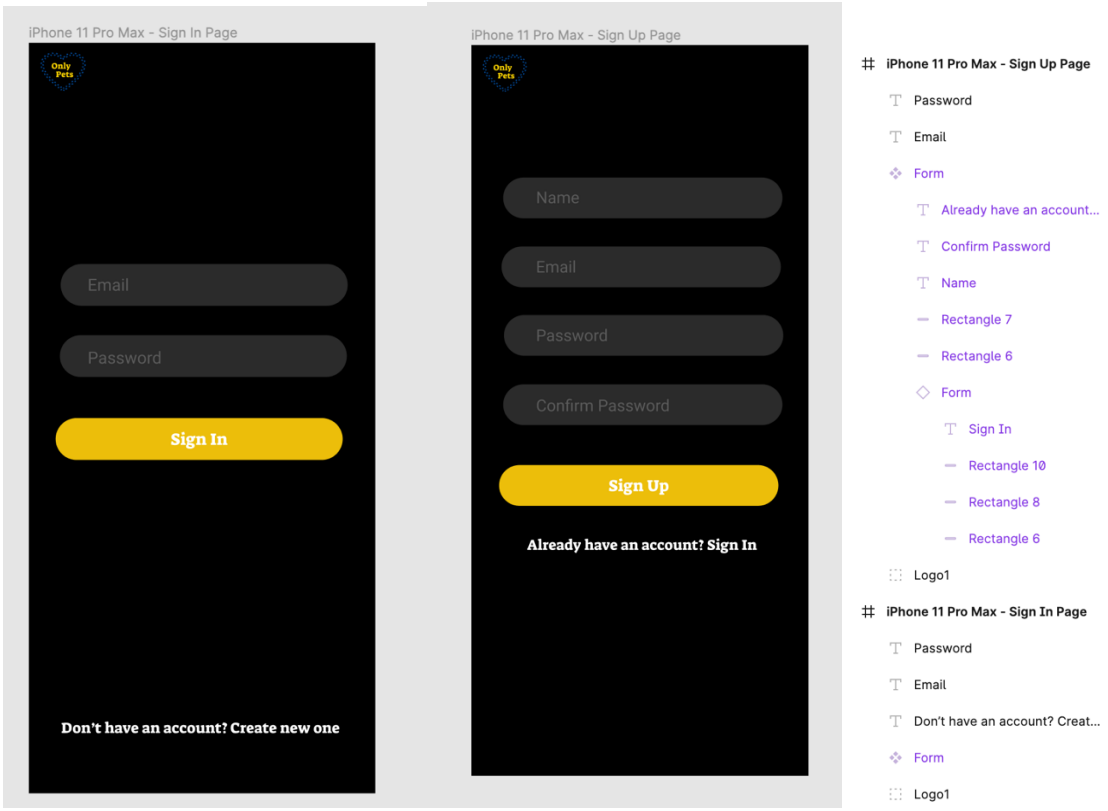


Рисунок 1.8 – Прототип додатку у Figma

При відкритті додатку користувач побачить сторінку для входу у додаток. У випадку, якщо профіль ще не створено, відображено кнопку з посиланням на сторінку створення нового профілю.

Почнемо розробку додатку. Середовищем для розробки було обрано Visual Studio Code.

Для початку потрібно встановити всі необхідні залежності. Перша – це watchman, що потрібна для слідкування за змінами у файловій системі. Далі встановимо Xcode для розробки додатку. Після цього треба встановити cocoapods, які необхідні для керування залежностями в проекті. Ініціалізуємо проект з допомогою команди npx react-native init projectName і перевіримо, чи працює проект, виконавши команди npx react-native start для запуску Metro, а в іншій вкладці терміналу – npx react-native run-ios для запуску симулятора.

На даному етапі ми маємо бачити початкову сторінку з дефолтними стилями. Після цього треба підготовити структуру проекту, створити необхідні файли.

Створимо файл index.js з допомогою якого ініціалізуємо проект, та передаємо йому компонент App.js, де власне будемо рендерити всі інші компоненти додатку.

Всі файли, зокрема, компоненти, хелпери, константи, будуть зберігатися в папці src/. Такі дані, як шрифти чи картинки зберігаються в папці assets/icons, Глобальні стилі- у папці styles/. Запити на сервер, інтерсептори, хелпери – це все зберігається у папці api/. Компоненти та скріни зберігаються у папках components/ та screens/ відповідно.

Отже, коли користувач відкриває додаток, відображаються або сторінка для входу/реєстрації, або ж головна сторінка. Для перевірки того, чи авторизований користувач, потрібно дістати токен з storage і перевірити, чи він дійсний. Якщо так, то користувач авторизований і має доступ до профілю, якщо ні – користувач має авторизуватися. Задля цього створено token-based authentication. Коли користувач авторизується, з бекенду повертаються два токена, один з них refreshToken. У випадку, коли перший токен стає недійсний, за допомогою middleware перевіряємо,

що якщо користувач робить запит на захищений шлях, і його токен не є дійсним, відправляємо refreshToken, щоб оновити токен і робимо перевірку. На бекенді робиться верифікація токена і підтвердження на його оновлення. Якщо все добре і токен існує, віддаємо на клієнт новий токен з оновленим часом дійсності. Якщо ж ні, тоді треба заново зайти в акаунт.

Сторінка для входу складається з двох текстових полів для вводу пошти та паролю та кнопки для підтвердження. Також є перехід на сторінку для реєстрації. При натисканні на кнопку, робимо перевірку на валідність даних. Вимоги - валідна пошта, для якої використовується стандартний регулярний вираз та пароль, що має містити мінімум 4 символи. Якщо введена невалідна пошта або пароль – відображається тост повідомлення про помилку. Якщо користувача знайдено з такою поштою – його успішно авторизовано.

Сторінка для реєстрації схожа на попередню, але ще має додаткові поля для вводу ім’я та підтвердження паролю. Додана додаткова валідація на паролі, оскільки вони мають бути однаковими, та для імені – воно обов’язкове. Після цього робимо запит на реєстрацію. Для того, щоб робити http запити, використовується бібліотека axios.

Для отримання введених користувачем даних та відображення їх у профілі, робимо GET запит в компоненті профіля. Задля того, щоб зберігати отримані від бекенда дані в одному місці, та відображати їх, навіть якщо користувач перезавантажив додаток, а також управляти ними, використовується бібліотека redux, зокрема, redux toolkit. Конфігуруємо так званий глобальний стор, та створюємо редьюсери для користувача та окремо для моделі тварини, оскільки це є окрема сутність.

Тобто коли робиться запит на отримання якихось даних, ми викликаємо action (дію), і обробляємо її відповідь у спеціальних функціях - редьюсерах.

На головній сторінці відображається ім’я користувача та його картинка. При натисканні на картинку ми переходимо до редагування профілю. Тут є можливість редагувати ім’я та пошту, а також вибрати стать та додати номер телефону. Також є можливість завантажити картинку з галереї.

Можливість вийти з акаунту також знаходиться в налаштуваннях.

При переході на профіль тварини, користувач або бачить форму для додавання, або вже існуючий профіль з заповненими полями. Тут робимо перевірку, що якщо немає ніяких даних про тварину, тобто її профіль ще не створено, показуємо відповідний текст “ Створити” і форму для заповнення. Якщо профіль тварини вже існує, тоді відображаємо “Редагувати” і робимо запит на редагування профілю.

Для того, щоб діставати певні дані зі стору, та користуватися ними в окремих компонентах, використовуємо селектори. Тобто це функції геттери, які дозволяють отримувати певну частину стейту, замість того, щоб звертатися до усього стейту напряму.

Переходимо до сторінки календаря. По суті, це певна agenda, де можна переглядати календар, як такий, та додавати нові події, нагадування, та переглядати їх. Для власників тварин це особливо корисно, адже можна записувати, коли буде наступний візит до ветеринару, або ж коли треба вакцинуватися.

Якщо ще не створено ніяких подій, користувач буде бачити календар, та підсвічену поточну дату, а при натисканні на будь-яку дату – блок з текстом, що подій ще не додано.

Для реалізації календаря використовується бібліотека react-native-calendars. Отже, в даному компоненті ініціалізуємо локальний стан для зберігання отриманих подій від бекенду, та для відображення модального вікна при натисканні на кнопку створення нової події. Даний календар вимагає дані в певному форматі, об’єкт, в якому ключ – це дата події та значення – масив об’єктів з назвами події, та іншими кастомними полями, які потім можна відображати. Тому для цього треба робити додаткову обробку на клієнті, щоб передавати об’єкт у правильному форматі для календаря.

Дати відправляються на сервер та зберігаються в форматі UTC.

Для того, щоб відображати дати у певному форматі в часовому поясі користувача, використовуються функції для їх обробки.

Календар потребує певного формату дати, тому також робимо обробку.

Одна з основних частин додатку – це можливість нотифікувати користувачів про знайдених чи втрачених тварин. Користувачу потрібно перейти в певний розділ, де буде відображатися форма. Для підказки, над текстовим полем для вводу відображено статичний текст, в якому вказуються всі деталі , що можна вказати для даного повідомлення, щоб іншим легше було розуміти, для чого воно створено. В даній формі існує валідація для введеного тексту, якщо воно коротше, ніж 5 символів, кнопка “Продовжити” буде неактивною.

Після того, як повідомлення додано, та при натисканні на кнопку “Продовжити”, користувач переходить до сторінки з мапою. В цьому компоненті є декілька кейсів. Перший з них – це саме коли користувач хоче створити нове повідомлення. В такому випадку, при натисканні на будь-яку точку на мапі, отримуємо її координати та записуємо у локальний стан. Якщо користувач водить по мапі, або натискає на іншу точку, ми перезаписуємо стан. Тільки після підтвердження створення повідомлення, ті координати, що були обрані, відправляються разом з іншими даними на бекенд.

Для зручності, на сторінці з повідомленнями також є інша вкладка – перегляд історії всіх створених повідомлень користувача. Там відображається блок з текстом, датою створення та кнопка для перегляду на мапі. Це є другий кейс. У даному випадку, при натисканні на перегляд на мапі, користувач переходить до мапи, та відбувається анімація з переходом до конкретної точки. Мапа в даному кейсі вже не доступна для створення нових точок, тільки у форматі view-only.

Компонент мапи пов’язаний з компонентом повідомлень. Проте, коли користувач переходить на мапу з головної сторінки, в нього є можливість переглядати всі точки, створені іншими, або самим користувачем. Мапа в данному випадку є view-only.

Для зручності, щоб відображати поточне місцезнаходження користувача на мапі, використовуємо додаткову бібліотеку.

Нам потрібно зробити запит на отримання геоданних тільки при монтуванні компонента, один раз, тому робимо це, використовуючи хук useEffect.

useEffect – це React хук, який викликається після того, як DOM змонтувався, або оновився. Викликаючи хук з порожнім масивом залежностей, ми кажемо, що хочемо, щоб дана функція виконалася тільки один раз при монтуванні компоненту. В даному випадку, потрібно дізнатись поточне місцезнаходження користувача, у випадку, якщо дана функція не заблокована. Після отримання геоданних, записуємо їх у локальний стан і відображаємо відповідно на карті.

У користувачів є можливість завантажити документи, наприклад, паспорт чи звіт про вакцинацію.

Для вибору файлів з телефону використовується бібліотека react-native-document-picker.

При завантаженні файлу робимо перевірку на розмір, та формуємо файл для відправки на сервер.

Після завантаження рендериться список доданих документів, натискаючи на які, можна подивитись їх зміст.

2.2 Розробка серверної частини

Для розробки серверної частини використовується Node.js.

У файлі app.js ініціалізуємо всі рути (шляхи), підключаємось до бази даних та стартуємо сервер. Обрана база даних – MongoDB. Отже, для початку роботи потрібно створити акаунт, та дістати URI, щоб підключитись до неї. Для зручності використовується MongoDB Compass, графічний інтерфейс, де можна переглядати та маніпулювати даними, колекціями. Додатково використовується фреймворк express.js, за допомогою якого можна ефективно створювати серверну частину додатку, API. Для валідації полей використовується express-validator.

Для обробки запитів на авторизацію, маніпуляцій з колекцією користувача та тварини використовуються різні API.

Модель користувача:

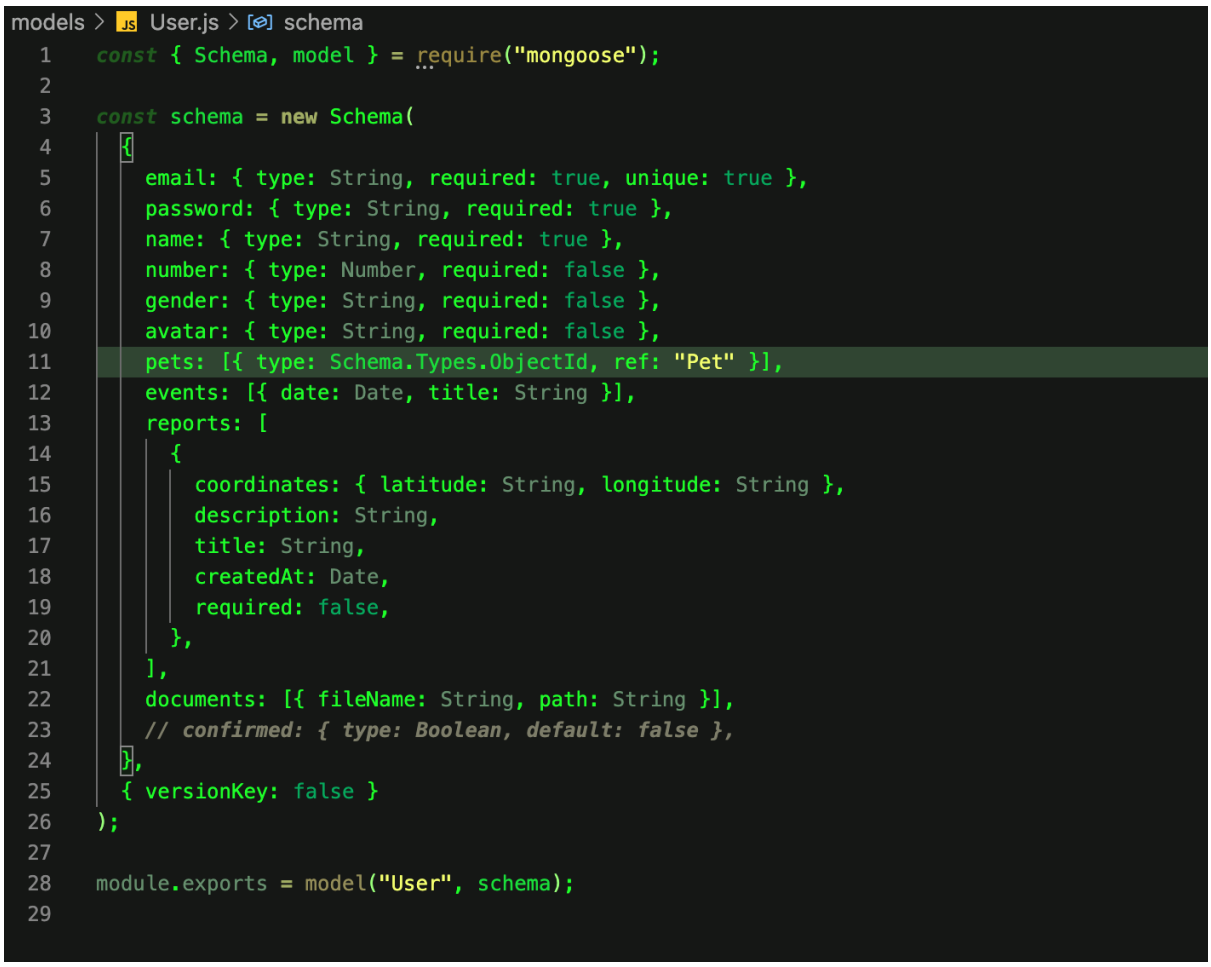


Рисунок 1.9 – Модель користувача

Коли користувач успішно заходить до акаунту, генеруються токени для подальшої валідації користувача.

До кожного запиту, що пов’язаний з отриманням чи створенням даних юзера, додаємо middleware для верифікації токена.

Якщо токен вже не дійсний, відправляємо запит на сервер для його оновлення.

Спочатку перевіряємо, чи є валідним токен, що надсилається з клієнта, якщо так - генеруємо новий, інакше користувачу треба авторизуватися.

Модель тварини:

models > Pet.js > ...

```
const { Schema, model } = require("mongoose");

const schema = new Schema(
  {
    sex: { type: String, required: false },
    name: { type: String, required: true },
    birthdate: { type: String, required: false },
    breed: { type: String, required: false },
    neutered: { type: String, required: false },
    allergy: { type: String, required: false },
    color: { type: String, required: false },
    type: { type: String, required: true },
    size: { type: String, required: false },
    avatar: { type: String, required: false },
  },
  { versionKey: false }
);

module.exports = model("Pet", schema);
```

Рисунок 1.10 – Модель тварини

Зв’язок всіх компонентів додатку:

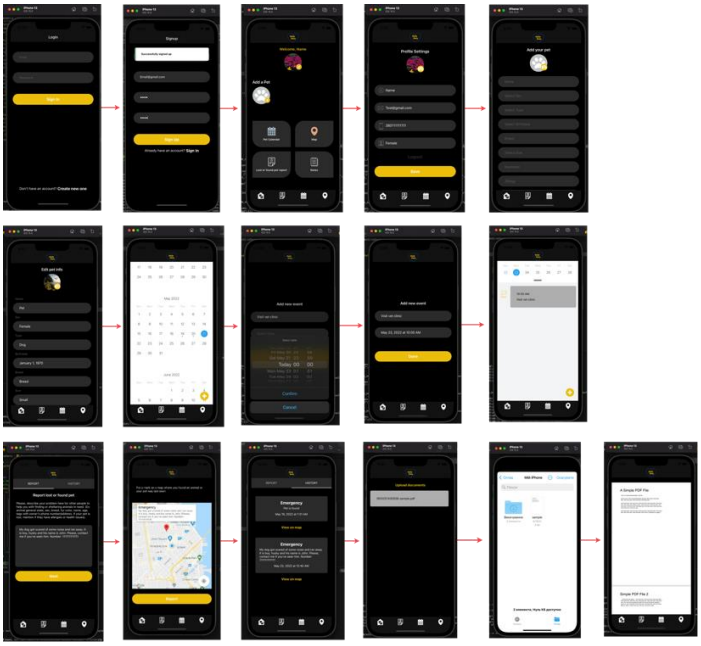


Рисунок 2.1 – Зв’язок елементів додатка

3. ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

При відкритті додатку відображається сторінка входу:

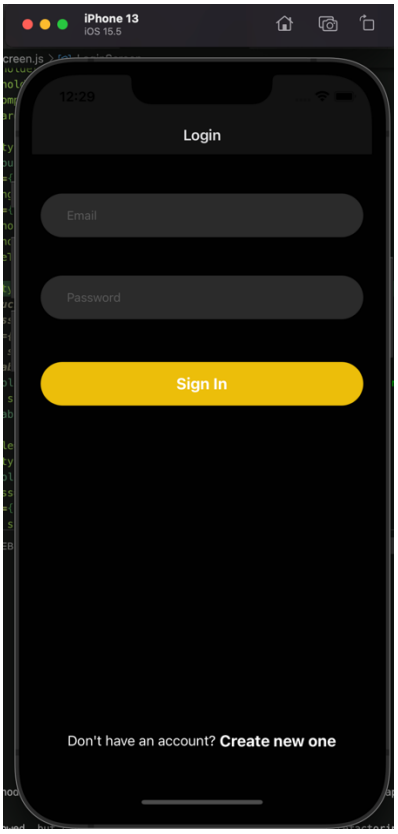


Рисунок 2.2 – Сторінка входу до акаунту

Якщо дані неправильного формату, відображаються помилки:

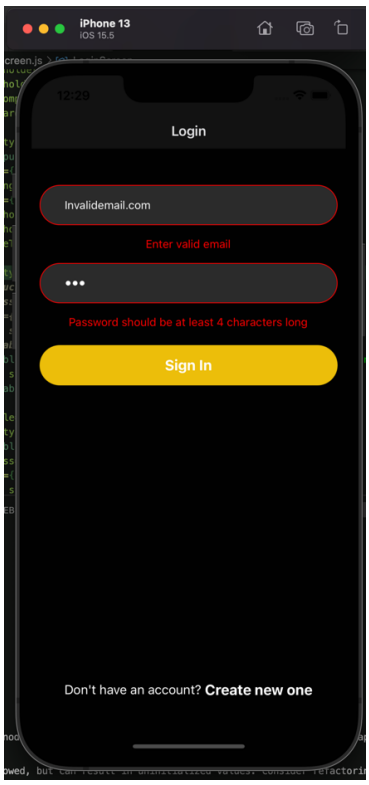


Рисунок 2.3 – Помилки при вході

Створення нового акаунту:

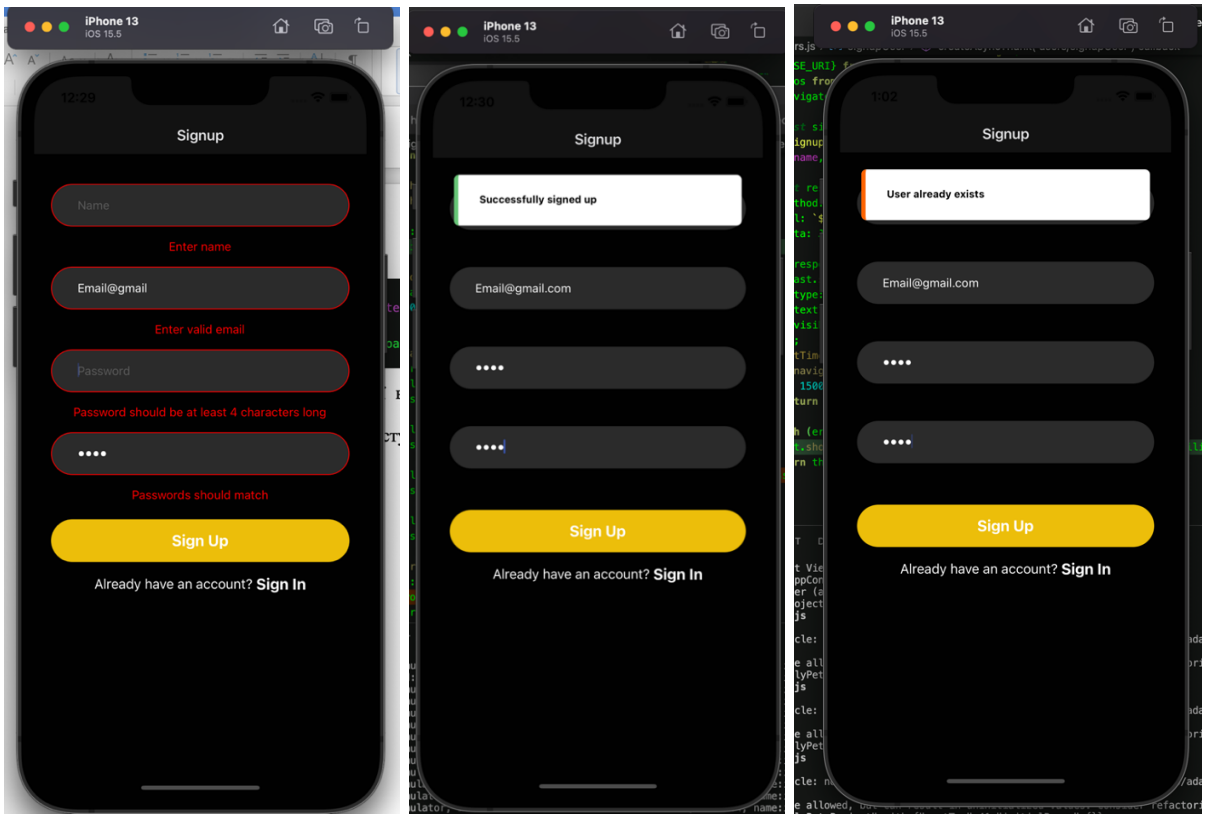


Рисунок 2.4 – Створення нового акаунту

Після входу відображається головна сторінка:

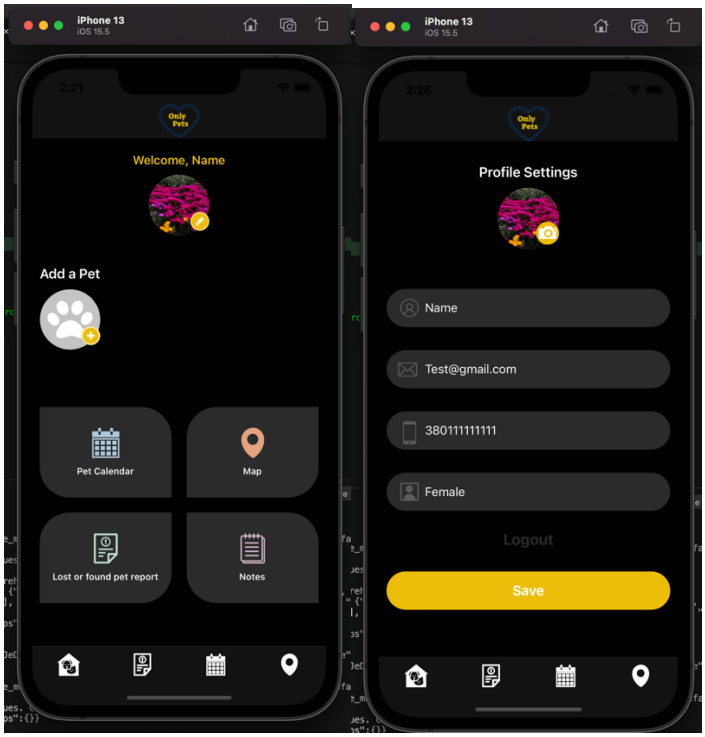


Рисунок 2.5 – Головна сторінка з переходом у налаштування профілю

Профіль тварини:

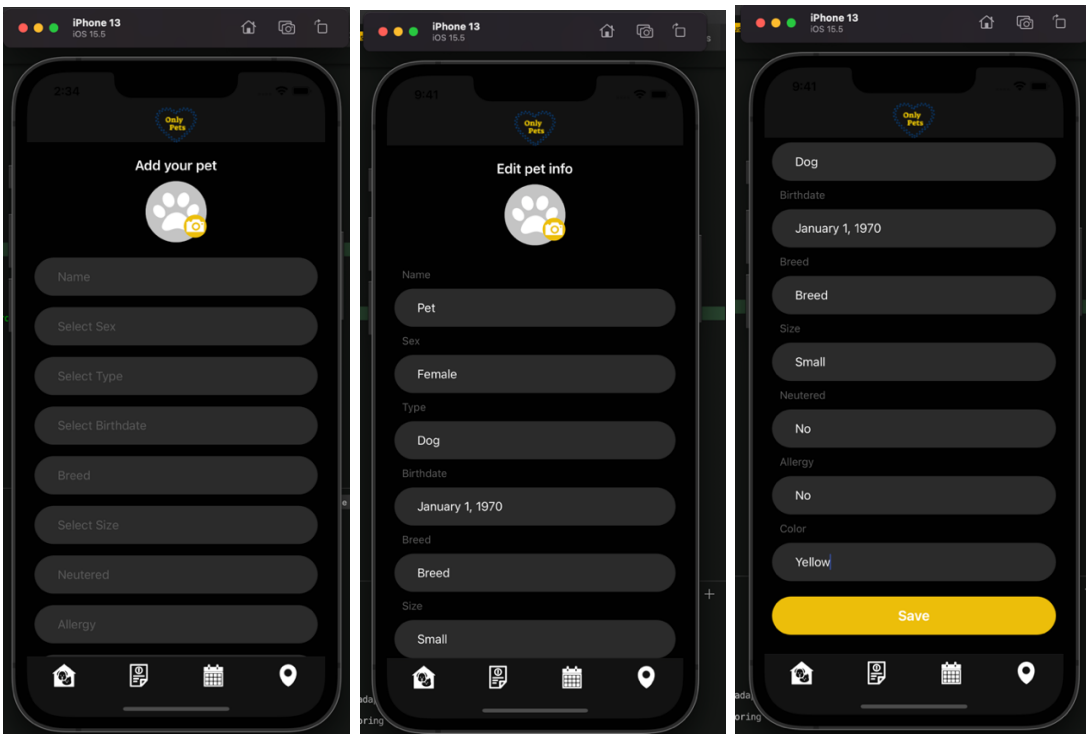


Рисунок 2.6 – Сторінка профілю тварини

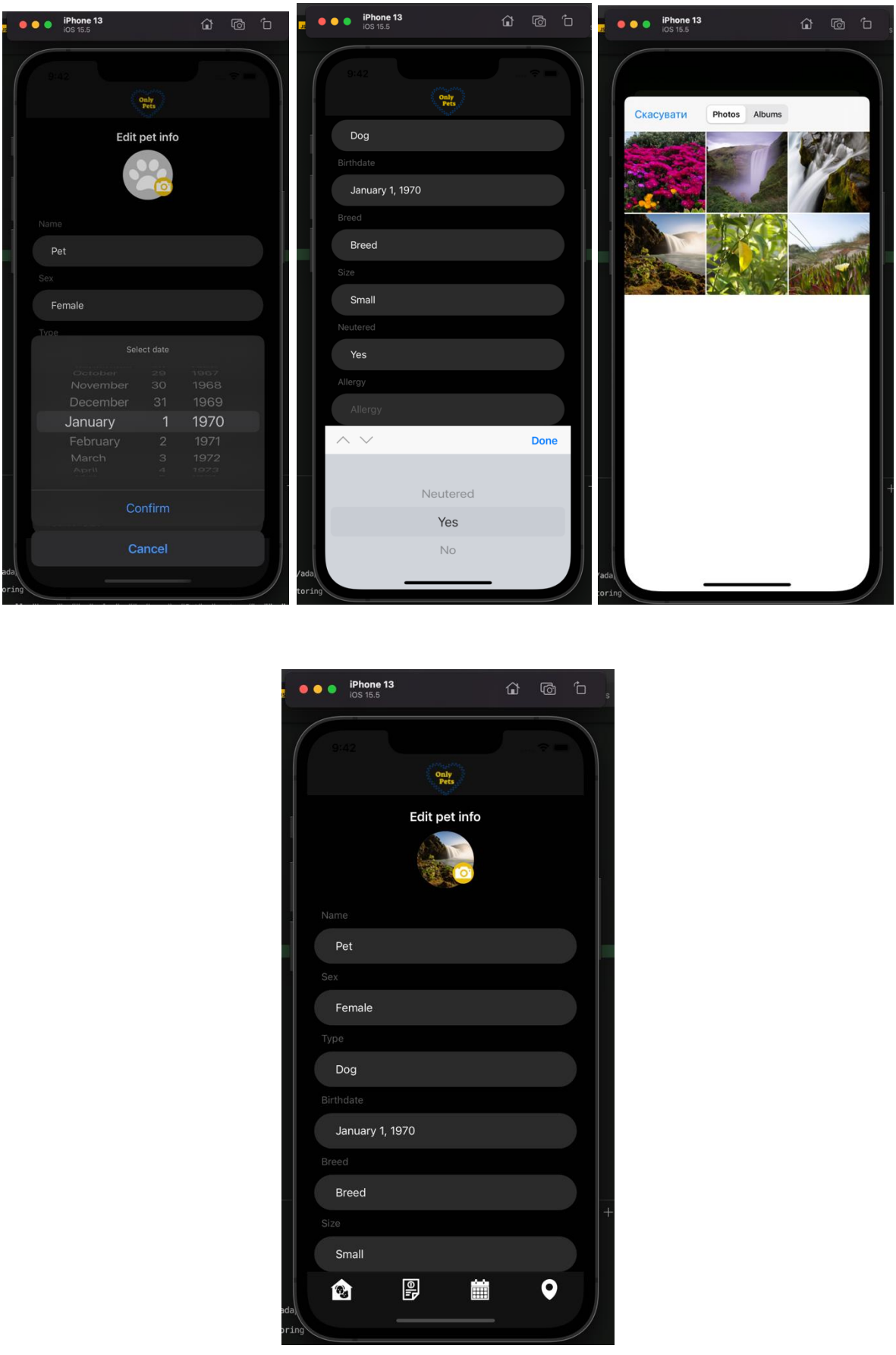


Рисунок 2.7 – Редагування профілю тварини

Сторінка з календарем:

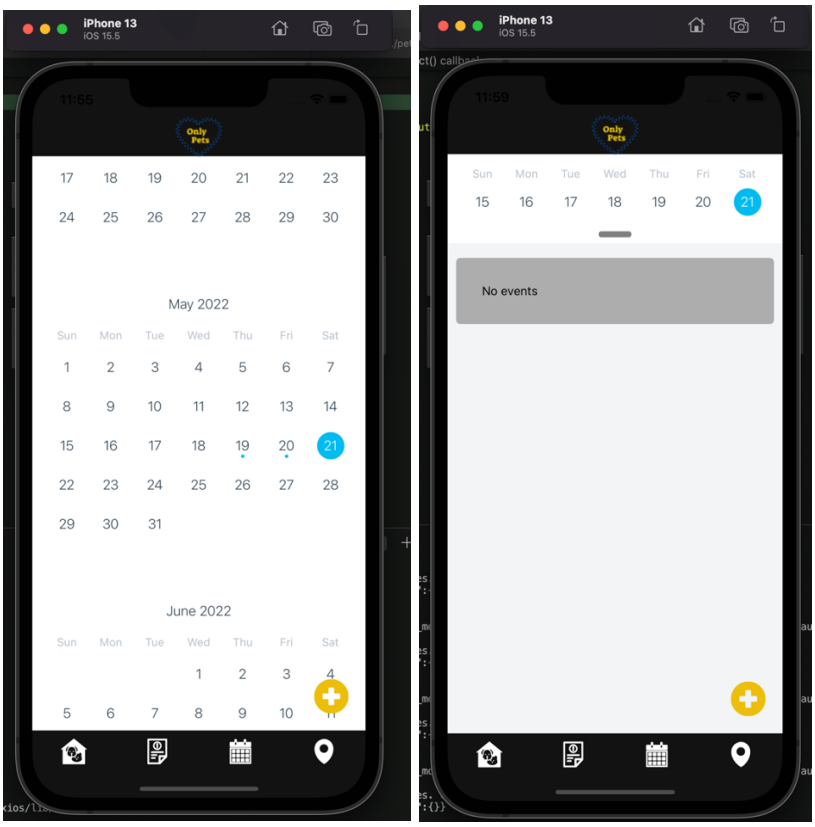


Рисунок 2.8 – Початкова сторінка календаря

Створення нової події:

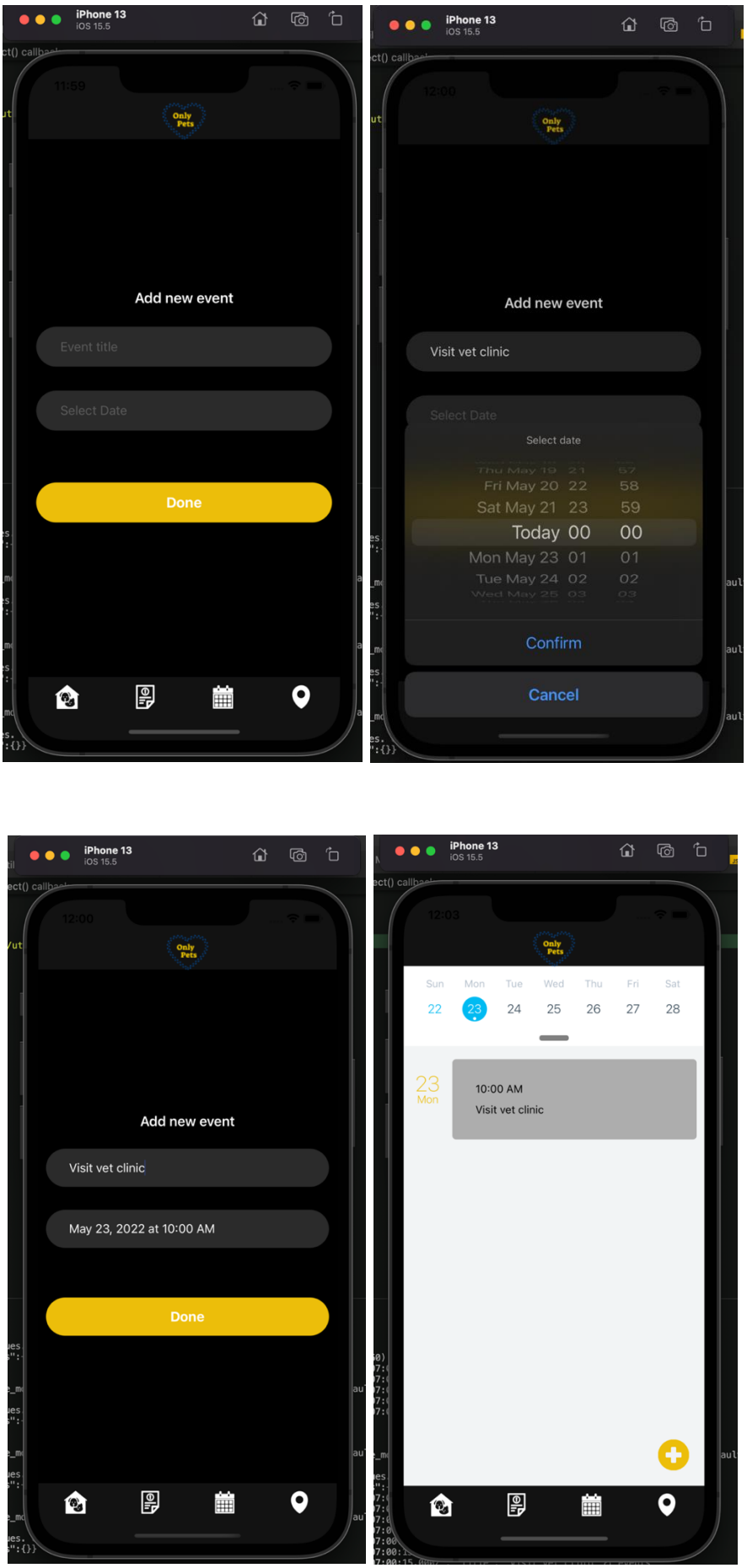
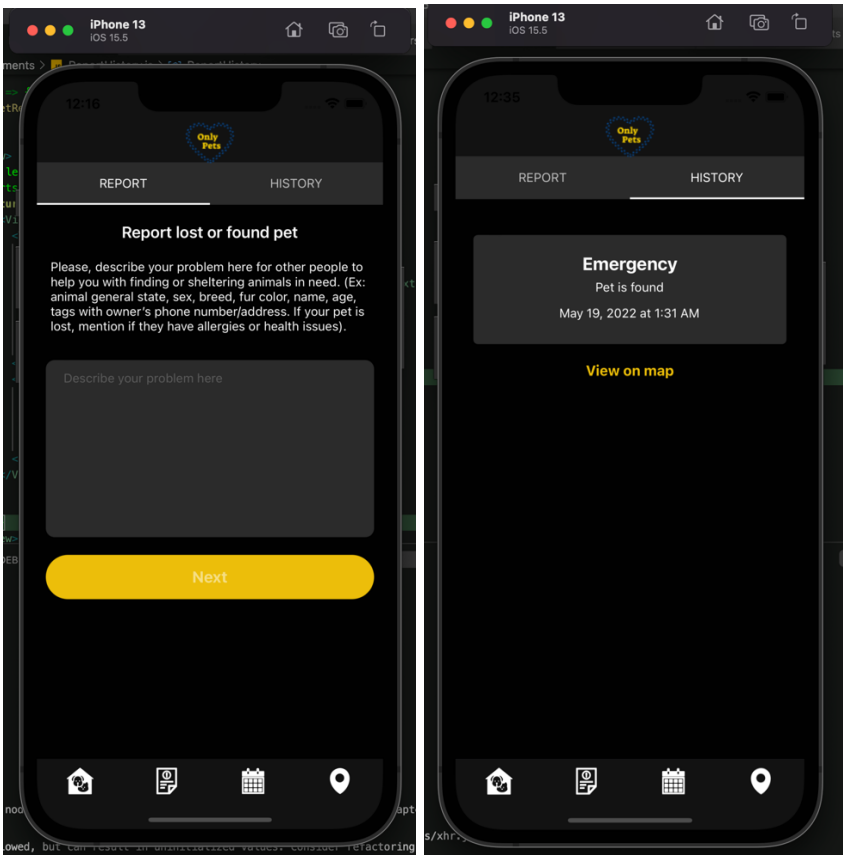


Рисунок 2.9 – Додавання нової події

Сторінки для створення та перегляду повідомлень:



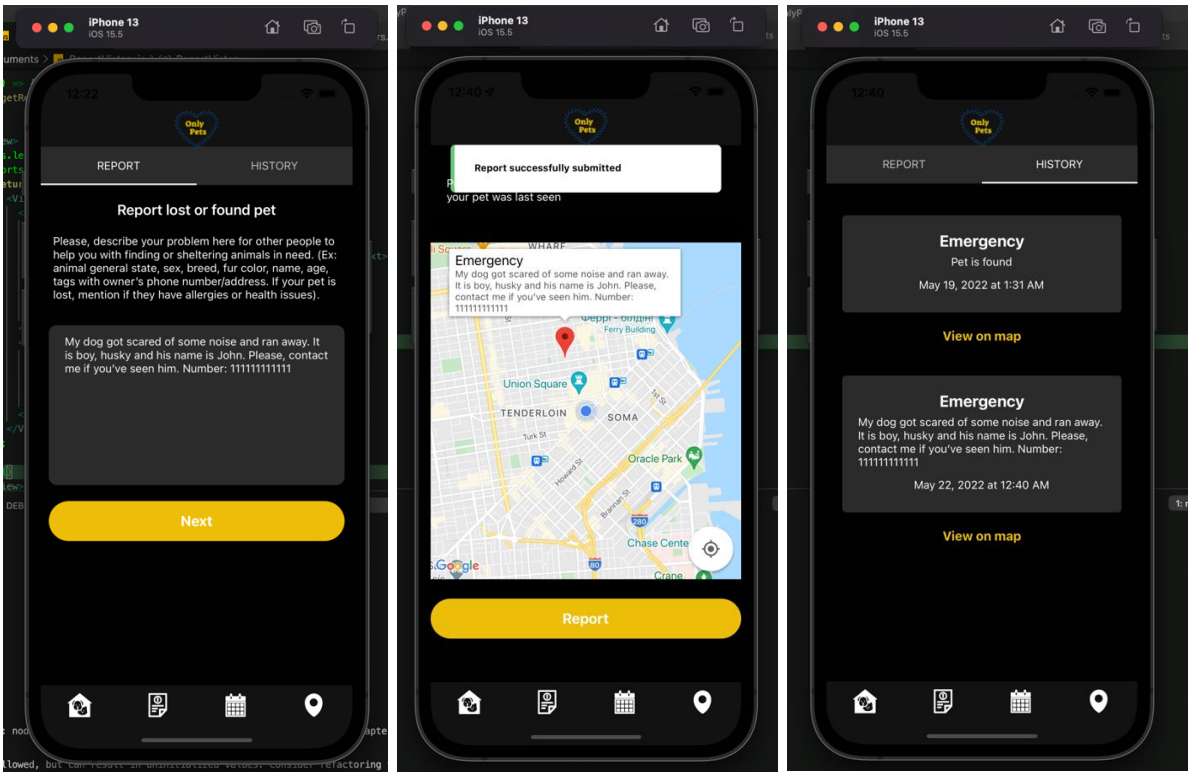


Рисунок 2.10 – Створення та перегляд повідомлень

Перегляд мапи:

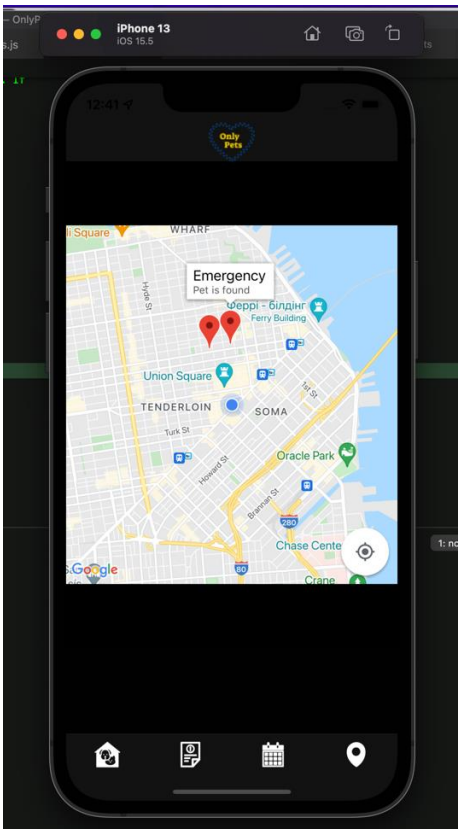


Рисунок 3.1 – Мапа

Завантаження та перегляд документів:

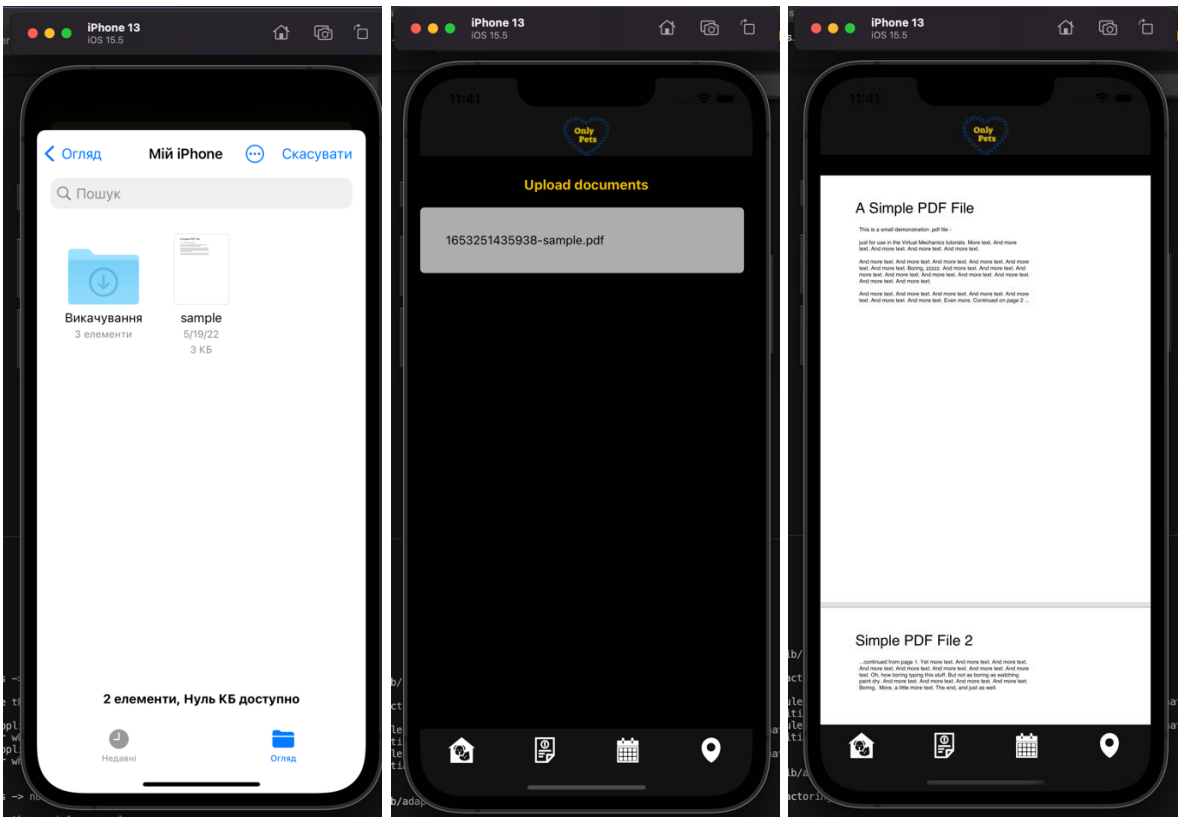


Рисунок 3.2 – Завантаження та перегляд документів

ВИСНОВКИ

Результатом виконання даної дипломної роботи є мобільний застосунок, що надає зручний інтерфейс як для власників тварин, так і для волонтерів, що допомагають безпритульним у знайденні домівки. У роботі описуються процеси виконання від аналізу наявних аналогів та актуальності даної теми до створення додатку.

1. Під час роботи було досліджено процес допомоги тваринам, знайдено шляхи для створення більш ефективного розповсюдження інформації серед людей, що об'єднані однаковою метою. Було вирішено створити мобільний додаток, оскільки це більш зручний спосіб комунікації.

2. Було проведено аналіз програмних засобів реалізації та інструментів для розробки, що найкраще підходять для створення кросплатформенного додатку. В ході дослідження було обрано середовище розробки Visual Studio Code, для розробки та тестування додатку – платформу Xcode. Для створення прототипу додатку було використано платформу Figma. Для розробки клієнтської частини використано фреймворк React Native, а для серверної - Node.js, база даних - MongoDB та git для контролю версій проекту.

3. Визначено основні вимоги та елементи інтерфейсу, розроблено прототип мобільного додатку, що демонструє графічний інтерфейс

Подальші перспективи даного додатку:

1. Розширений функціонал для покриття всіх основних потреб користувачів

2. Створення архітектури для зв'язку користувача з іншими, можливість додавати друзів, та переписуватись у чаті

3. Відображення головних новин, посилання на актуальні проекти, фонди

ПЕРЕЛІК ПОСИЛАНЬ

1. Robert Martin Clean Code: A Handbook of Agile Software Craftsmanship - https://www.amazon.com/Clean-Code-Handbook-Software-Craftsmanship/dp/0132350882
2. React Native Official Documentation - https://reactnative.dev/docs/environment-setup
3. Bonnie Eisenman Learning React Native: Building Native Mobile Apps with JavaScript – Second edition - https://www.amazon.com/Learning-React-Native-Building-JavaScript/dp/1491989149
4. Ethan Brown Web Development with Node and Express: Leveraging the JavaScript Stack – Second edition - https://www.amazon.com/Web-Development-Node-Express-Leveraging/dp/1491949309
5. Eric Masiello Mastering React Native - https://www.amazon.com/Mastering-React-Native-Eric-Masiello/dp/1785885782
6. Emilio Rodriguez Martinez React Native Blueprints: Create eight exciting native cross-platform mobile applications with JavaScript - https://www.amazon.com/React-Native-Blueprints-cross-platform-applications/dp/1787288099
7. Mateusz Grzesiukiewicz Hands-On Design Patterns with React Native: Proven techniques and patterns for efficient native mobile development with JavaScript – First edition - https://www.amazon.com/Hands-Design-Patterns-React-Native-ebook/dp/B07FSRYMH8
8. Dan Ward React Native Cookbook: Recipes for solving common React Native development problems – Second edition - https://www.amazon.com/React-Native-Cookbook-Industry-Development-ebook/dp/B079TVH9C3
9. Nader Dabit React Native in Action – First edition - https://www.amazon.com/React-Native-Action-Nader-Dabit/dp/1617294055

10. Frank Zammetti Practical React Native: Build Two Full Projects and One Full Game using React Native – First edition - https://www.amazon.com/Practical-React-Native-Build-Projects/dp/1484239385
11. Eric Bush JavaScript Applications with Node.js, React, React Native and MongoDB: Design, code, test, deploy and manage in Amazon AWS - https://www.amazon.com/dp/0997196661?tag=uuid10-20
12. Eric Bush Node.js, MongoDB, React, React Native Full-Stack Fundamentals and Beyond - https://www.amazon.com/dp/0997196688?tag=uuid10-20
13. Ethan Holmes, Tom Bray Getting Started with React Native - https://www.amazon.com/dp/1785885189?tag=uuid10-20
14. Redux Toolkit Official Documentation - https://redux-toolkit.js.org/introduction/getting-started
15. React Native Overview - https://geekflare.com/react-native-for-mobile-app/

ДОДАТКИ

Додаток А

middleware > auth.js > ...

```
1   const jwt = require("jsonwebtoken");
2   const config = require("config");
3
4
5   // middleware for verifying client token, added to each protected route
6   module.exports = (req, res, next) => {
7     try {
8       const token = req.headers["x-access-token"];
9       const { userId } = req.query;
10      const decodedToken = jwt.verify(token, config.get("jwtRefreshTokenSecret"));
11      const id = decodedToken._id;
12      if (userId && userId !== id) {
13        throw "Invalid user id";
14      } else {
15        next();
16      }
17    } catch (err) {
18      res.status(401).json({
19        message: "Token has expired",
20        error: new Error("Invalid request!"),
21      });
22    }
23  };
24
```

Рисунок 3.3 – Middleware для перевірки токена

routes > auth.routes.js > router.post("/register") callback

```
18  //api/auth/register
19  router.post(
20    "/register",
21    [
22      check("email", "Invalid email").isEmail(),
23      check("password", "Min password length 4 symbols").isLength({ min: 4 }),
24      check("name", "Name should not be empty").notEmpty(),
25      check("confirmPassword", "Min password length 4 symbols")
26        .isLength({ min: 4 })
27        .custom(async (confirmPassword, { req }) => {
28          const password = req.body.password;
29          if (password !== confirmPassword) {
30            throw new Error("Password must match");
31          }
32        }),
33    ],
34    async (req, res) => {
35      try {
36        const errors = validationResult(req);
37        if (!errors.isEmpty()) {
38          return res.status(400).json({
39            errors: errors.array(),
40            message: "Invalid register data",
41          });
42        }
43        const { email, password, name } = req.body;
44
45        const candidate = await User.findOne({ email });
46        if (candidate) {
47          return res.status(400).json({ message: "User already exists" });
48        }
49
50        const hashedPassword = await bcrypt.hash(password, 12);
51
52        await User.create({
53          email,
54          password: hashedPassword,
55          name
56        });
57
58        res.status(201).json({
59          message: "User created",
60        });
61      } catch (e) {
```

Рисунок 3.4 – обробка запиту реєстрації користувача

```
// api/pet/edit
router.post("/edit", auth, async (req, res) => {
  try {
    const { id, petAvatar, ...rest } = req.body;
    const pet = await Pet.findById(id);
    if (!pet) {
      return res.status(400).json({ message: "Pet not found" });
    }
    const updatedPet = await Pet.findByIdAndUpdate(
      { _id: pet.id },
      { avatar: petAvatar, ...rest, $set: { avatar: petAvatar } },
      { upsert: true, new: true }
    );

    res.status(200).json({
      message: "Pet updated",
      pets: updatedPet,
    });
  } catch (e) {
    res.status(500).json({ message: "Something went wrong. Try again" });
  }
});
```

Рисунок 3.5 – Редагування профілю тварини

```
src > components > profile > JS EditProfile.js > ...
21  export const EditProfile = () => {
22    const user = useSelector(selectUser);
23    const avatar = useSelector(selectUserAvatar);
24    const [values, setValues] = useState({
25      name: user?.name || '',
26      email: user?.email || '',
27      number: user?.number?.toString() || null,
28      gender: user?.gender || 'None',
29      avatar: avatar || '',
30    });
31    const dispatch = useDispatch();
32    const navigation = useNavigation();
33
34    const handleChange = (name, value) => {
35      setValues({...values, [name]: value});
36    };
37
38    const handleSubmit = () => {
39      dispatch(editUser(values));
40      navigation.navigate('HomeScreen');
41    };
42
43    const handlePicker = async () => {
44      const result = await launchImageLibrary();
45      if (result.errorCode) {
46        Toast.show({type: 'error', text1: result.errorCode});
47      }
48      if (result.assets) {
49        handleChange('avatar', result.assets[0].uri);
50      }
51    };
52
53    const handleLogout = () => {
54      dispatch(logout());
55    };
```

Рисунок 3.6 – Редагування профілю користувача на клієнті

src > components > map > Map.js > ...

```
export const Map = props => {
  const isReport = !!props.route.params?.report;
  const reportsCoordinates = useSelector(selectReports);
  const [position, setPosition] = useState({
    latitude: 50.450001,
    longitude: 30.523333,
    latitudeDelta: 0.001,
    longitudeDelta: 0.001,
  });
  const [marker, setMarker] = useState(
    !isReport && reportsCoordinates.length
      ? reportsCoordinates
      : {
          coordinates: null,
          title: 'Emergency',
          description: props.route.params?.report || '',
        },
  );

  const mapRef = React.createRef();

  const dispatch = useDispatch();

  useEffect(() => {
    Geolocation.getCurrentPosition(pos => {
      const coordinates = pos.coords;
      setPosition({
        latitude: coordinates.latitude,
        longitude: coordinates.longitude,
        latitudeDelta: 0.0421,
        longitudeDelta: 0.0421,
      });
    }).catch(err => {
      Toast.show({type: 'error', text1: 'Something went wrong'});
    });

    if (!isReport) {
      dispatch(getReports());
    }
  }, []);

  useEffect(() => {
    if (mapRef) {
      changeRegion();
```

Рисунок 3.7 – Компонент мапи

src > redux > store.js > ...

```
import {configureStore} from '@reduxjs/toolkit';
import {persistStore, persistReducer} from 'redux-persist';
import AsyncStorage from '@react-native-async-storage/async-storage';
import {combineReducers} from 'redux';
import {userSlice} from './user/user';
import {petSlice} from './pet/pet';
import {axiosMiddleware} from '../api/interceptors';

const persistConfig = {
  key: 'root',
  storage: AsyncStorage,
};

const appReducer = combineReducers({
  user: userSlice.reducer,
  pet: petSlice.reducer,
});

const rootReducer = (state, action) => {
  if (action.type === 'user/logout') {
    state = {};
  }
  return appReducer(state, action);
};

const persistedReducer = persistReducer(persistConfig, rootReducer);

export const store = configureStore({
  reducer: persistedReducer,
  middleware: getDefaultMiddleware =>
    getDefaultMiddleware({
      serializableCheck: false,
    }).concat(axiosMiddleware),
});
export const persistor = persistStore(store);

```

Рисунок 3.8 – Конфігурація Redux стор

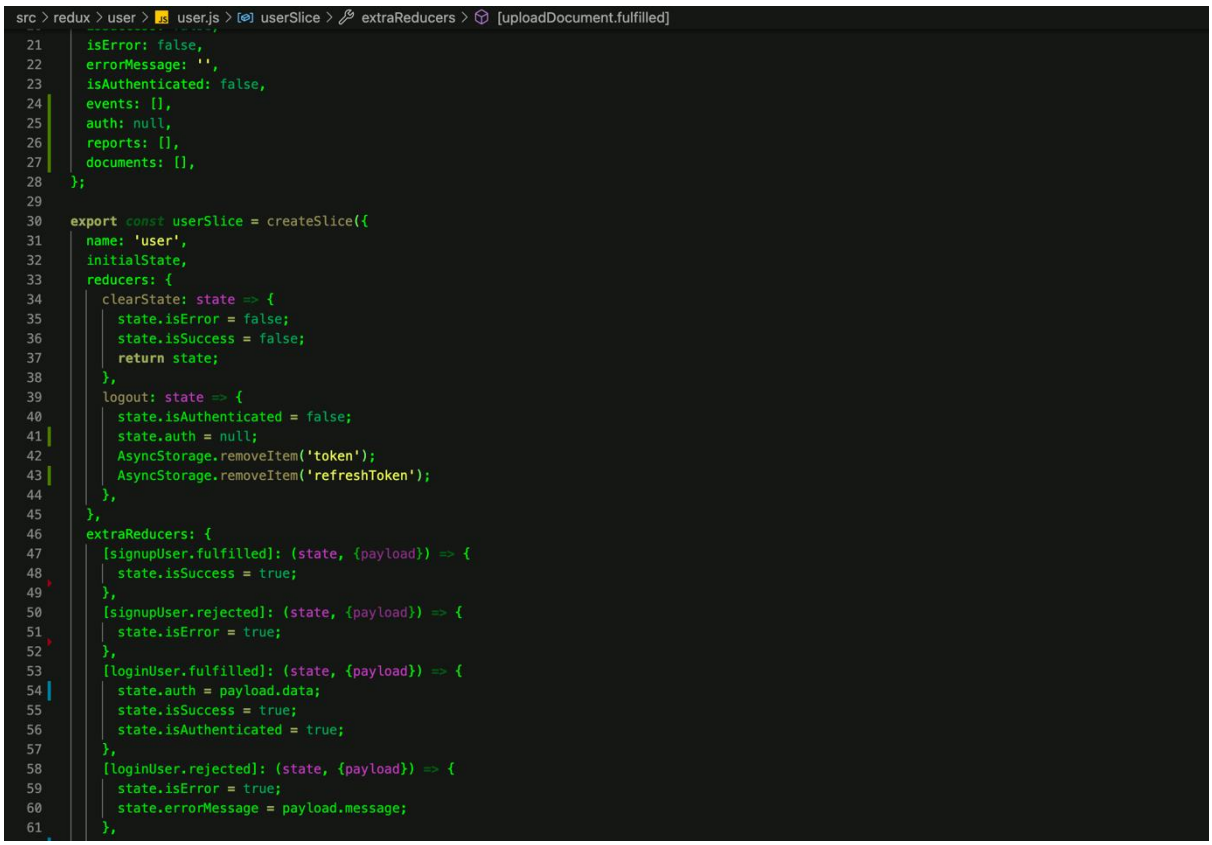


Рисунок 3.9 – Редьюсер користувача

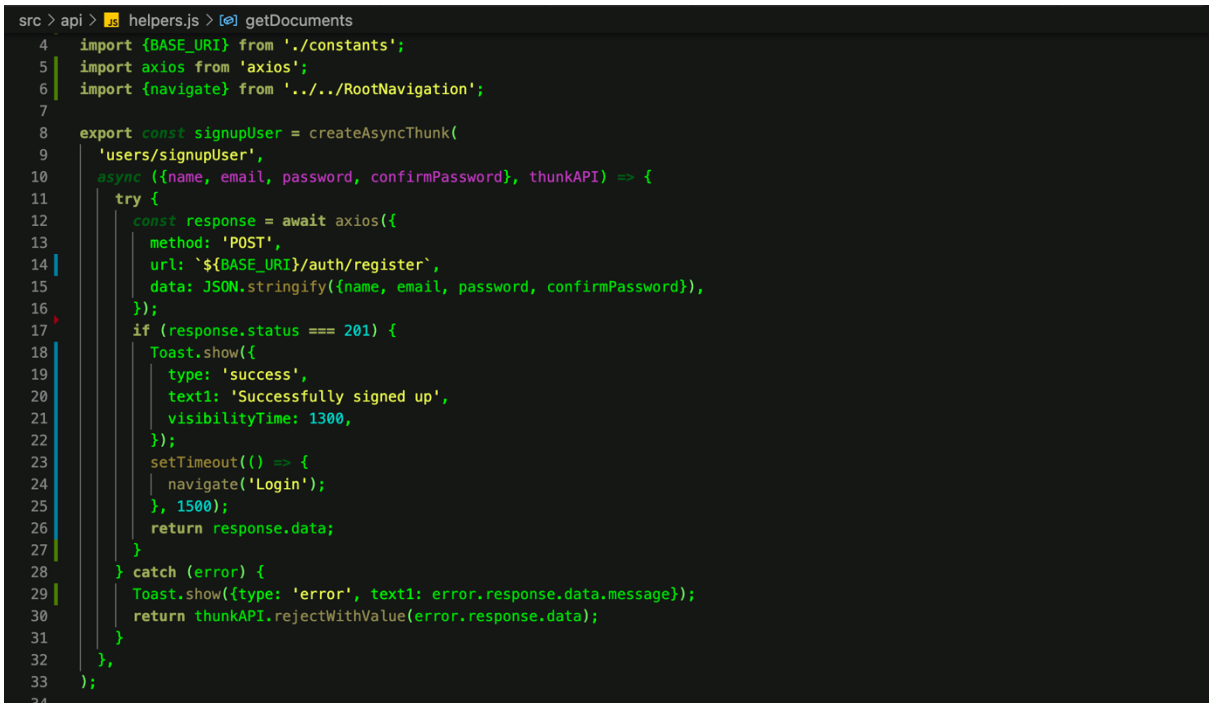


Рисунок 3.10 – Запит на реєстрацію користувача

src > selectors > selectors.js > ...

```
import {createSelector} from '@reduxjs/toolkit';

export const selectAuth = state => state.user.isAuthenticated || false;
export const selectUser = state => state.user.userData.user || null;
export const selectUserAvatar = createSelector(
  selectUser,
  state => state?.avatar || null,
);

export const selectPet = state => state.pet?.petData?.pets || null;
export const selectEvents = state => state.user.events || [];
export const selectToken = state => state.user.auth || null;
export const selectReports = state => state.user.reports || [];
export const selectDocuments = state => state.user.documents || [];
```

Рисунок 4.1 – Redux селектори

src > api > interceptors.js > interceptor > axios.interceptors.response.use() callback

```
const interceptor = store => {
  axios.interceptors.request.use(
    async config => {
      const token = getToken(store);
      if (token) {
        config.headers['x-access-token'] = token;
      }
      return config;
    },
    error => {
      return Promise.reject(error);
    },
  );
  axios.interceptors.response.use(
    res => {
      return res;
    },
    async err => {
      const originalConfig = err.config;
      if (err.response) {
        if (err.response.status === 401) {
          try {
            const rs = await store.dispatch(refreshToken());
            const {accessToken} = rs.payload;
            await setToken(accessToken);
            axios.defaults.headers.common['x-access-token'] = accessToken;
            return axios(originalConfig);
          } catch (_error) {
            if (_error.response && _error.response.data) {
              return Promise.reject(_error.response.data);
            }
            return Promise.reject(_error);
          }
        }
        if (err.response.status === 403 && err.response.data) {
          return Promise.reject(err.response.data);
        }
      }
      return Promise.reject(err);
    },
  );
};
```

Рисунок 4.2 - Інтерсептор

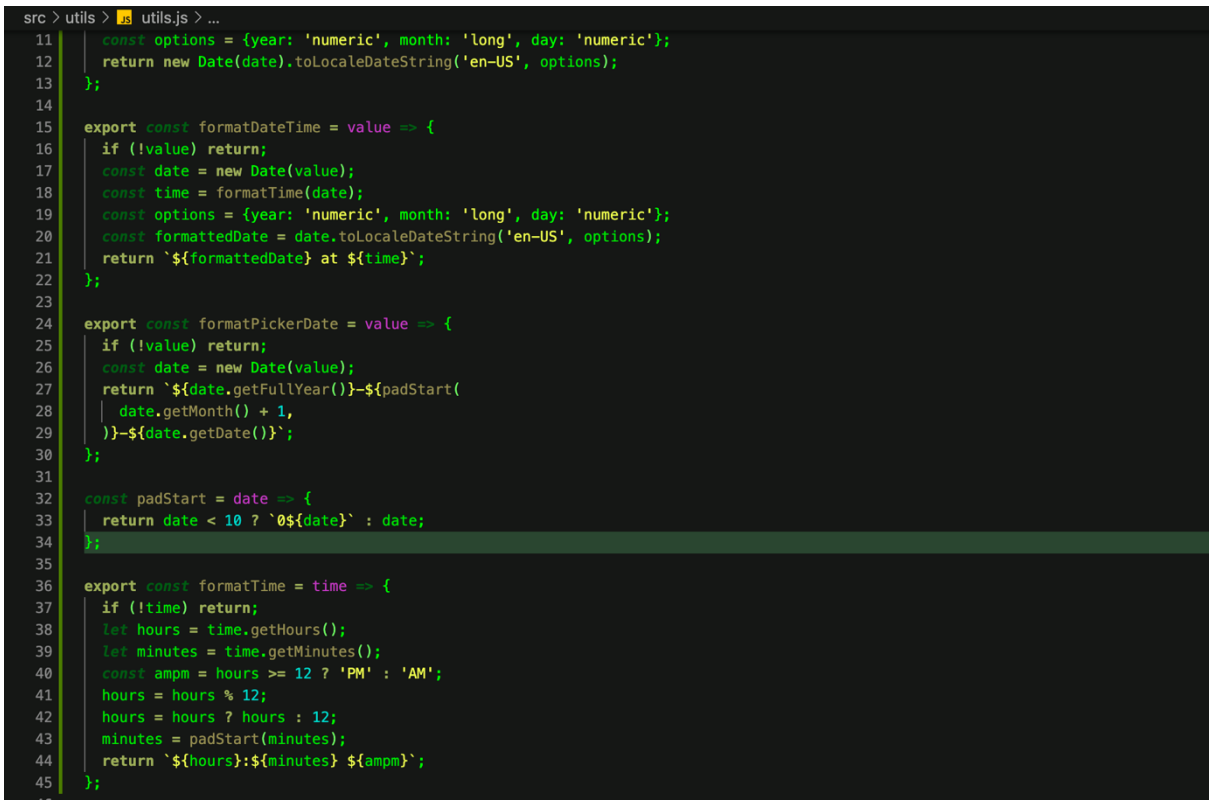


Рисунок 4.3 – Хелпери для обробки та форматування дат

Додаток Б

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ ДЛЯ ВЛАСНИКІВ ТВАРИН НА ОСНОВІ NODE.JS ТА REACT NATIVE»

Виконала студентка 5 курсу
Групи ППЗ-51
Шевцова Тетяна Ігорівна
Керівник роботи
Жебка Вікторія Вікторівна, доцент кафедри

Київ – 2022

1

Мета, об’єкт та предмет дипломної роботи

Мета дипломної роботи – покращення процесу допомоги тваринам в режимі реального часу з допомогою застосування мобільного додатку, написаного на React Native та Node.js.

Об’єкт дослідження – процес більш ефективної та оперативної допомоги тваринам, можливість ведення щоденнику тварини для власників тварин

Предмет дослідження – програмне забезпечення, що сприятиме більш ефективній допомозі тваринам

2

Ключові проблеми галузі

Метою даного застосунку є надання зручного інтерфейсу та функціоналу для власників тварин, що включає планування, наприклад, візитів до ветеринару, створення профілю тварини, завантаження документів, комунікація з іншими користувачами.

Інша проблема, що вирішується, це допомога безпритульним тваринам.

Основною комунікацією щодо знайдених тварин, яким потрібен прихисток, чи втрачених домашніх тварин, є соцмережі. Проте маємо розуміти, що це займає час, поки заявку на публікацію оформлять, підтвердять, та опублікують.
В додатку є можливість в разі знаходження такої тварини, додавати цю інформацію та відображати відразу на мапі. Отже, інші користувачі теж зможуть її бачити, та відповідно реагувати.

3

Завдання бакалаврської роботи

1. Проаналізовано проблему та сучасні методи її вирішення
2. Проаналізовано аналоги, створено порівняльну таблицю
3. Досліджено програмні засоби для реалізації додатку
4. Створено діаграми
5. Розроблено клієнтську та серверну частини додатку
6. Протестовано готовий додаток
7. Визначено подальші вдосконалення додатку

4

Технічні завдання

- Створення профіля користувача
- Створення профіля тварини
- Створення календарю-agenda
- В календарі має бути можливість додавати, видаляти та переглядати події
- Створення мапи
- Розробка функціоналу додавання нових точок з певною інформацією до мапи
- Інформування інших користувачів у випадку зникнення чи знаходження тварини
- Створення функціоналу для додавання документів

5

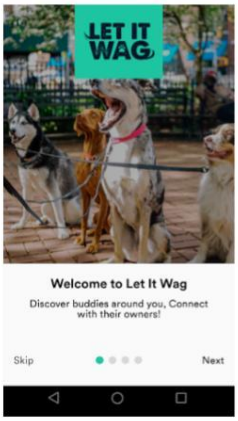
Аналоги

Переваги:

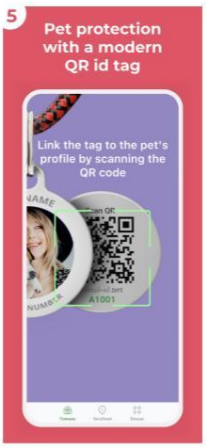
- швидка допомога тваринам локальними організаціями та волонтерами
- допомога у знаходженні прихистків
- послуги з вигулу тварин
- благодійна допомога різним фондам та притулкам

Недоліки:

- наявність багів, через що неможливо протестувати наявний функціонал



Аналоги



Переваги:

- дуже зручний додаток для власників тварин
- можливість додавати нагадування до календаря
- відстежування ваги тварини
- магазин, в якому можна придбати мікрочіп або жетон
- мапа, по якій можна відстежити загубленого пухнастика з допомогою мікрочіпу
- завантаження документів

Недоліки:

- немає функції для сповіщення про знайдених безпритульних тварин, що не мають мікрочіпу

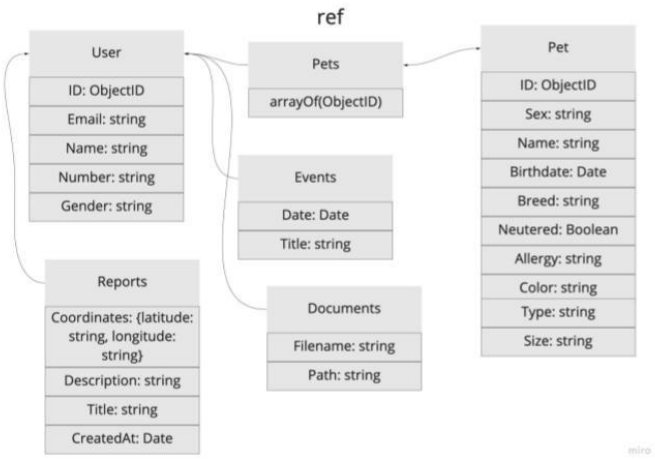
Порівняльна таблиця

Показник	Animal ID	Let it Wag	OnlyPets
Платформи	Android, iOS, Web	Android, iOS, Web	Android, iOS
Створення профілю	+	+	+
Створення профілю тварини	+	-	+
Оперативна допомога тваринам в режимі реального часу	-	+	+
Допомога з прилаштуванням тварин	+	+	+
Наявність календарю для ведення звіту по стану здоров'я тварини	+	-	+
Наявність мапи з геолокацією тварини	+	+	+
Наявність безкоштовної версії	+	+	+
Можливість завантажити документи	+	-	+

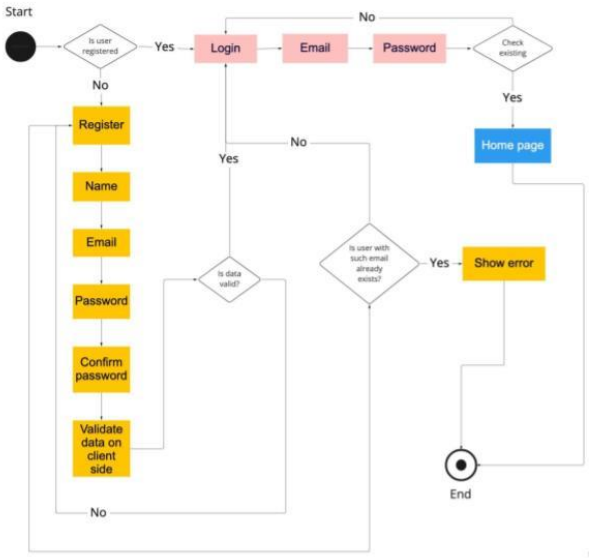
Програмні засоби реалізації

Схема бази даних

Документо-орієнтована СУБД MongoDB

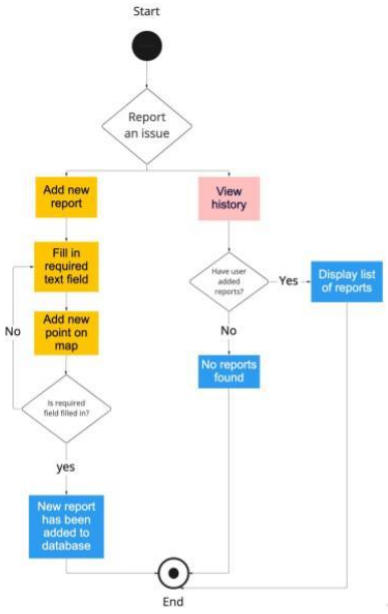


UML діаграма діяльності аутентифікації користувача



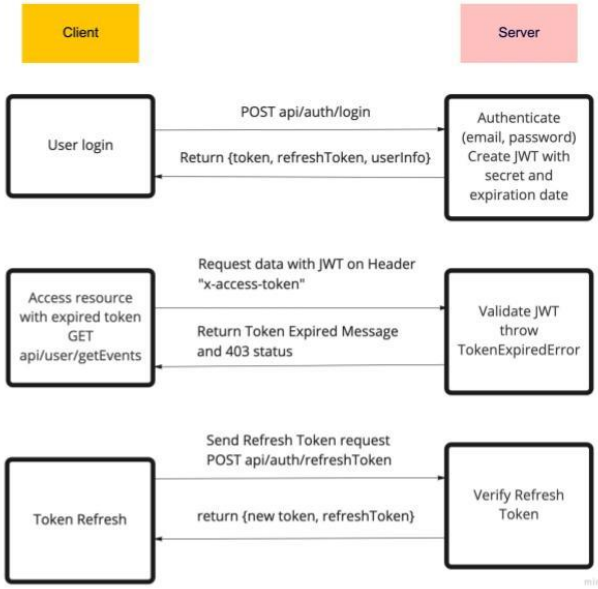
11

UML діаграма діяльності створення нового повідомлення



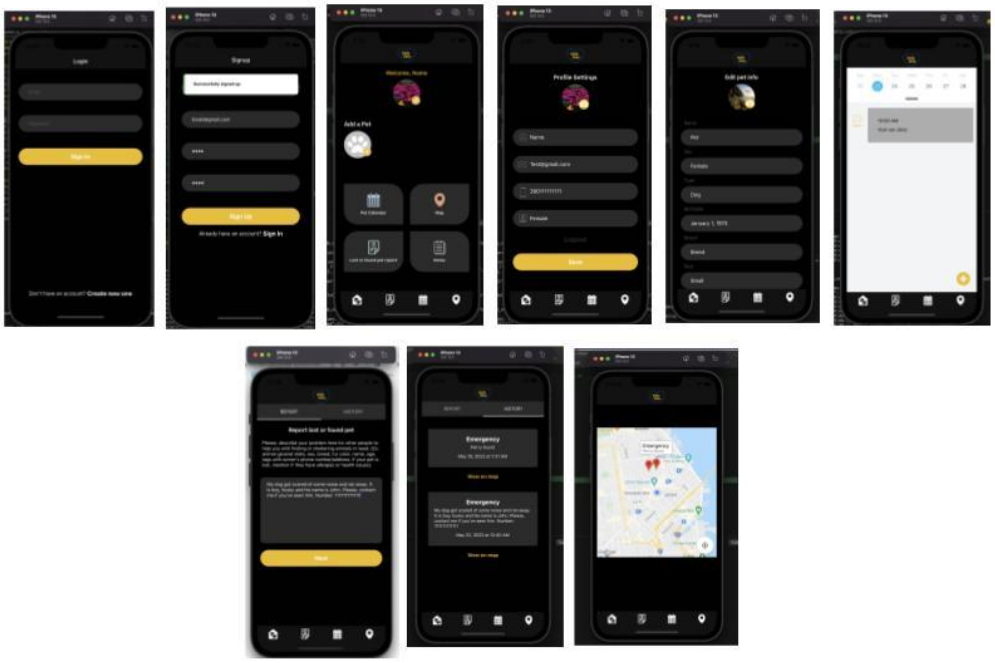
12

Схема Token-based авторизації



13

Готовий додаток OnlyPets



Апробація результатів дослідження

1. Шевцова Т.І. Розробка цифрової платформи для власників тварин на основі Node.js та React.js / Т.І. Шевцова // Застосування програмного забезпечення в інфокомунікаційних технологіях: Матеріали науково-технічної конференції, ДУТ, м. Київ, 20 квітня 2022 року.

Висновки

1. Було досліджено процес роботи волонтерських організацій, допомогу безпритульним та втраченим домашнім тваринам, актуальність створення додатку для розповсюдження інформації, комунікації таких організацій
2. Проаналізовано програмні засоби реалізації, інструменти для створення швидкого додатку та зручного інтерфейсу для користувачів
3. Визначено основні вимоги та елементи інтерфейсу, розроблено прототип мобільного додатку, що демонструє графічний інтерфейс

Подальші перспективи даного додатку:

1. Розширений функціонал для покриття всіх основних потреб користувачів
2. Створення архітектури для зв’язку користувача з іншими, можливість додавати друзів, та переписуватись у чаті
3. Відображення головних новин, посилання на актуальні проекти, фонди

Дякую за увагу!