МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
Факультет електроніки та інформаційних технологій

Кафедра комп'ютерних наук

Кваліфікаційна робота бакалавра

«ІНФОРМАЦІЙНЕ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ ДЛЯ ОРГАНІЗАЦІЇ РОБОТИ ЛІКАРІВ ВЕТЕРИНАРНОЇ МЕДИЦИНИ ТА ЇХ КЛІЄНТІВ»

Здобувач освіти гр. ІН – 81 Дар'я БІЛОУС

Науковий керівник Борис КУЗІКОВ
старший викладач, кандидат техн. наук

Завідувач кафедри Анатолій ДОВБИШ
професор, доктор технічних наук

СУМИ 2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютерних наук

Затверджую_____
Зав. кафедрою Довбиш А.С.
"_____" _____2022 р.

ЗАВДАННЯ

до кваліфікаційної роботи бакалавра

Студентки четвертого курсу, групи ІН-81 спеціальності спеціальності 122 - Комп'ютерні науки денної форми навчання Білоус Дар'ї Дмитрівни.

Тема: Інформаційне та програмне забезпечення системи для організації роботи лікарів ветеринарної медицини та їх клієнтів

Затверджена наказом по СумДУ
№_____ від _____2022 р.

Зміст пояснювальної записки: 1) Аналіз предметної області; 2) Моделювання та проектування інформаційної системи; 3) Практична реалізація інформаційної системи; 4) Висновки;

Дата видачі завдання "_____"_____2022 р.
Керівник випускної роботи _____ Кузіков Б.О.
Завдання прийняв до виконання _____ Білоус Д.Д.

РЕФЕРАТ

Записка: 65 стор., 57 рис., 1 табл., 2 додатки, 11 джерел.

Об’єкт дослідження — процеси взаємодії лікарів ветеринарної медицини та їх клієнтів, приватна ветеринарна практика.

Мета роботи — створенню Web-додатку для автоматизації взаємодії лікарів та клієнтів, вирішення проблеми доступу клієнтів до медичної документації тварин, проблеми бронювання запису на прийом, проблеми передачі медичних призначень тварини між лікарями.

Методи дослідження — методи аналізу документальної інформації

Результати — був проведений аналіз існуючих аналогів, крім того було досліджено актуальність проблеми, здійснено вибір засобів реалізації. Було виконано програмну реалізацію інформаційної системи.

ІНФОРМАЦІЙНА СИСТЕМА, ВЕТЕРИНАРНА МЕДИЦИНА, ОНЛАЙН БРОНЮВАННЯ

ЗМІСТ

ВСТУП

Життя людини з домашнім улюбленцем складно уявити без відвідування ветеринара. Навіть якщо тварина нічим не хворіє, вона має проходити профілактичні огляди. Коли справа стосується вибору лікаря, власники починають питати порад у знайомих та читати відгуки про лікарів та клініки в інтернеті.

Тож перша з проблем - пошук клініки. Це не важко зробити у великому місті, але не всі ветеринари обирають працювати у закладі. Деякі віддають перевагу заснуванню власної справи, а в маленьких містах клінік може не бути взагалі, і єдиний варіант працювати - приватний кабінет. Крім того спеціалістів з деяких сфер ветеринарної медицини може не бути у невеликій клініці. Але як зручно ознайомитися з усіма доступними приватними ветеринарами, якщо не кожен з них має можливість замовити розробку власного сайту, та не всі кабінети відмічені на картах?

Друга проблема - пам’ятати усе, що лікар сказав. Ветеринарні клініки знаходяться на різних етапах розвитку сьогодні. У деяких досі дають лише папірець, на якому від руки написані рекомендації, назва ліків та як їх давати тварині. Інші вже видають призначення більш організовано, це може бути шаблонізований роздрукований лист. У першому випадку клієнт може не зрозуміти почерк лікаря, і доведеться телефонувати у клініку. В обох варіантах клієнт може загубити або випадково зіпсувати отриманий лист. Все закінчується дзвінком до клініки, що може нервувати і клієнтів і працівників.

На сьогоднішній день користуватися інтернетом можна не лише заради розваг. Інтернет економить час і гроші для людей, робить виконання деяких задач зручнішими. Тому онлайн-покупки, віддалені зустрічі та навіть віддалений найм набули популярності.

Як інтернет може стати корисним для приватних ветеринарів і володарів тварин? Ветеринарам було б добре зберігати власні дані клієнтів і результати обстеження їх тварин не на папері. Непогано ще надати змогу клієнтам

переглядати історію своїх прийомів самостійно, замість витрати часу на дзвінки виду “Я був у вас два тижні тому, забув назву пігулок, потрібних для мого кота”. Бачити розклад своїх записів на день в одному місці теж краще, ніж шукати серед неупорядкованих папірців хто наступний.

Крім того попит на онлайн-запис на прийом зріс за останні роки. Отже, пропозиція онлайн-послуги запису на прийом до ветеринара теж не буде зайвою. Чому ж онлайн запис до ветеринара полегшує життя клієнтів? Система онлайн-бронювання має дві ключові переваги.

По-перше, це простота: клієнту легше записатися на прийом для своєї тварини онлайн. Зручність стає на перше місце, коли справа доходить до онлайн-бронювання з будь-якого місця та в будь-який час. За допомогою системи онлайн-бронювання клієнти можуть записатися на зустріч посеред ночі – якщо це для них найзручніше. Статистика показує, що 55% клієнтів вважають за краще бронювати зустріч у неробочий час [9].

Друга перевага стосується безпосередньо лікаря або його помічника. Замість того, щоб завантажувати себе на передовій відповідями на телефонні дзвінки або відповідями на електронні листи та ручним бронюванням зустрічі з клієнтами, це можна вирішити онлайн. Це звільняє лікаря для більш цінного спілкування з клієнтами на практиці.

Виходячи із актуальності проблеми метою роботи буде автоматизація процесу взаємодії лікарів ветеринарної медицини і їх клієнтів шляхом розробки інформаційної системи. Для її реалізації необхідно буде виконати такі кроки:

- Аналіз визначеної предметної області
- Вивчення процесу взаємодії ветеринара і клієнта, виділення етапів що можна автоматизувати
- Проведення порівняльного аналізу сайтів-аналогів
- Вибір програмних засобів для реалізації додатку, його шаблон
- Розробка та реалізація моделі бази даних для додатку
- Розробка додатку

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд останніх досліджень і публікацій

Автоматизація робочих процесів у ветеринарній справі має суттєве значення. Це дозволяє зосередитися на найважливішому аспекті — турботі про тварин.

У сфері, яка розглядається, можна виділити дві основні сторони що взаємодіють: лікар ветеринарної медицини і клієнт. З боку лікарів перше, чого стосується автоматизація, це ведення документації. Традиційно медичні документи оформлялися в паперовому вигляді. Подібне було і у випадку з ветеринарною практикою. Записи ветеринарів про стан здоров’я тварини, сертифікати про вакцинацію, підписані власниками тварин, погодження процедур тощо. Усі медичні записи, написані на папері, потрібно зберігати в надійному місці, щоб можна було взяти їх у будь-який момент та поділитися ними із власником тварини або з іншим ветеринаром. Це означає, що лікар повинен забезпечити безпеку медичної документації (у разі пожежі, повені чи інших катастроф). Таких проблем можна уникнути, коли документація не паперова. З боку ж клієнтів зручно мати до неї доступ без відвідування клініки.

Нижче розглянуто переваги електронного зберігання документації:

- Покращена ефективність у роботі: друкують багато людей швидше, ніж пишуть від руки[10]. Робити нотатки під час прийому просто, і більше ніколи не виникне проблема розібрати що написано, адже так почерк не має значення. Рука більше не буде боліти від великої кількості писанини.
- Зменшення впливу на навколишнє середовище — електронне зберігання записів замість паперової документації дозволяє захистити навколишнє середовище. Це означає менше паперового сміття і майже відсутність друку, адже документацією можно поділитися через інтернет.

- Підвищена безпека медичної документації: зберігання важливих медичних даних тварин у системі захищає їх від знищення. Навіть у випадку катастрофи та зламаному комп’ютері всі дані залишаються цілими.
- Зменшення витраченого часу на непрямі обов’язки: якщо у клієнта буде доступ до документів власних тварин, він не буде відволікати лікаря дзвінками щоб взяти їх. Тож якщо постійний лікар тварини у відпустці це не є перешкодою для термінового відвідування іншого спеціаліста.

Можливість онлайн планування прийомів може допомогти збалансувати день як для ветеринарів, так і для їхніх клієнтів. Таке рішення дозволить клієнтам самостійно записатися на прийом заздалегідь. Онлайн планування зменшує адміністративне навантаження на самого лікаря або його допоміжний персонал.

Клієнти віддають перевагу онлайн-плануванню, оскільки вони можуть бачити одразу декілька доступних дат для запису, можуть зробити запис у неробочий час лікаря, можуть самостійно керувати своїми записами без потреби телефонувати та очікувати. Коли перед очима є всі можливі дні - легше обрати той, що підходить, аніж коли клієнту по телефону перераховують по одному варіанту и для кожного він повинен подивитися чи не перетинається цей день з його планами.

<u>Існують різні види бронювання :</u>

- *Фіксований розклад зустрічей.* Це популярний варіант, у якому призначається прийом у певний час, щоб кожен пацієнт знав, коли саме він відвідає ветеринара.
- *Хвильове планування.* Такий варіант передбачає планування від чотирьох до шести клієнтів на годину. Потім лікар надає послуги в порядку прибуття кожної людини (невелика жива черга).
- *Подвійне бронювання.* Це ефективна стратегія — бронювання одного запису для двох клієнтів одночасно — варіант для тих, хто хоче щоб пацієнт завжди

був готовий прийти та було менше “пустих вікон”. В такому варіанті лікар виділяє достатньо часу для двох пацієнтів, але якщо один з них не явиться - може приділити увесь час іншому.

Усі ці методи кращі, ніж відкритий робочий графік, який передбачає надання клієнтам певного проміжку часу, коли вони можуть прийти та відвідати ветеринара.

Можливість самостійно переглядати записи по своїм тваринам та керувати їх персональною інформацією є не менш корисною, ніж онлайн бронювання. Багато клінік наразі не надають такої можливості, адже в їх системах не передбачена роль для клієнта (володоря тварини). Зазвичай додає тварин і заповнює про них потрібну інформацію адміністратор закладу або сам приватний ветеринар. Таку процедуру доводиться виконувати кожній клініці, коли людина приходить до них вперше, та кожному приватному ветеринару в якого з’являється новий клієнт. Людині це може не дуже подобатись. Навіть якщо ветеринар має онлайн бронювання, то не всюди для нього треба реєструватися, а лише вказати номер і можливо електронну пошту.

У програмному забезпеченні, яке вимагає від людини спершу зареєструватися а потім робити онлайн бронювання візиту, гарним є те, що на момент приходу людини у клініку її основна інформація вже є в сисетмі. Кожен клієнт при першому зверненні повинен заповнити форму, щоб надати свою контактну інформацію, історію хвороби домашньої тварини та інші потрібні лікарю дані. Заповнення такої форми під час реєстрації зменшує кількість документів, необхідних до заповнення на місці. Така можливість також покращує задоволеність клієнта, оскільки вона зменшує кількість часу, який людина проводить у лікаря, перш ніж той приступає до діагностики.

Чому онлайн бронювання це зручно - клієнт може самостійно керувати своїм записом. Лікарю не потрібна зв'язуюча ланка у вигляді адміністратора та не потрібно самому відволікатись на дзвінки щоб перенести чи скасувати візит.

1.2 Існуючі програмні засоби для використання ветеринарами

Для проведення порівняльного аналізу було обрано декілька програмних забезпечень із аналогічним призначенням та схожою тематикою. Серед них є веб додатки, програмне забезпечення, яке потребує установки, і CRM система. Аналіз проведено за вісьмома критеріями, які можуть у сукупності дати загальну картину переваг і недоліків кожного додатку.

Перший критерій це потреба встановлення програмного забезпечення. На мій погляд відсутність даної потреби є плюсом, тому що у такому випадку не важлива операційна система користувача, додаток буде працювати у будь-якому браузері. PetMedico, Megaplan, Appointer (крім мобільної версії яка потребує установки) це веб додатки а Ветсофт-ветеринар і Андіаг десктопні.

Другий критерій наявність мобільної версії. Зручно у випадку коли треба щось подивитись а користувач не біля комп'ютера. PetMedico має адаптивну верстку (рис. 1.1), Megaplan та Appointer мають мобільні застосунки для операційних систем android та ios (рис. 1.2 та 1.3).

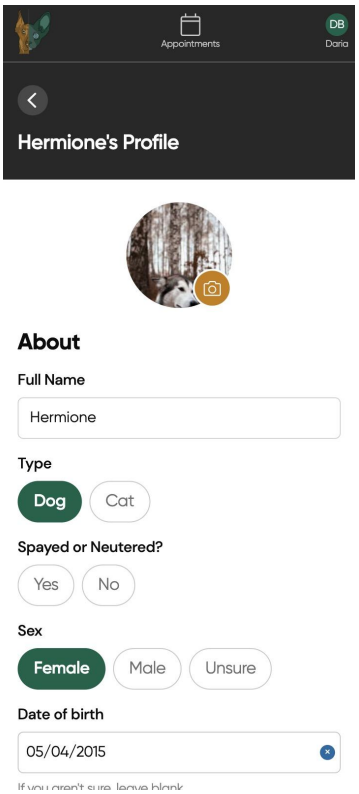


Рисунок 1.1 - вигляд додатку PetMedico на мобільному пристрої

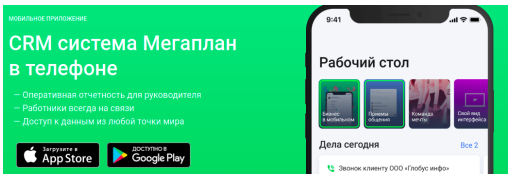


Рисунок 1.2 - мобільна версія Megaplan

Рисунок 1.3 - мобільна версія appointer

Програми ветсофт-ветеринар і андіаг не мають мобільних версій.

Третім критерієм є сучасний дизайн. Мова йде не лише про те що дизайн повинен бути красивим, а й зручним. Megaplan, Appointer і PetMedico можна віднести до застосунків із сучасним дизайном (Рис. 1.4, 1.5, 1.6) в той час як ветсофт-ветеринар і андіаг навряд чи (рис. 1.7 та 1.8).

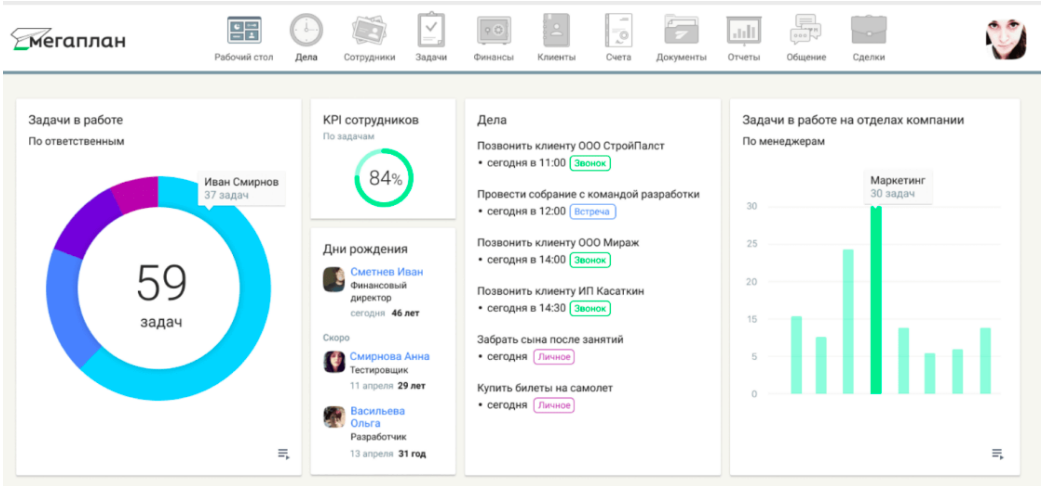


Рисунок 1.4 - дизайн Megaplan

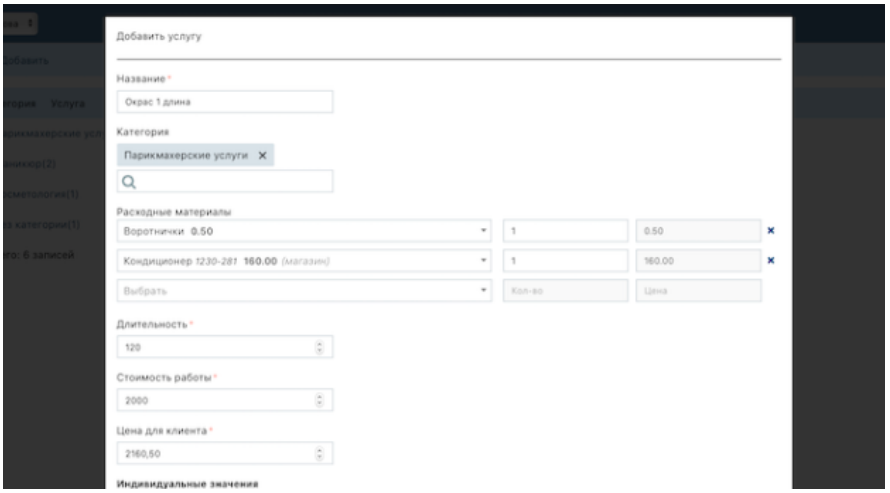


Рисунок 1.5 - дизайн Appointer

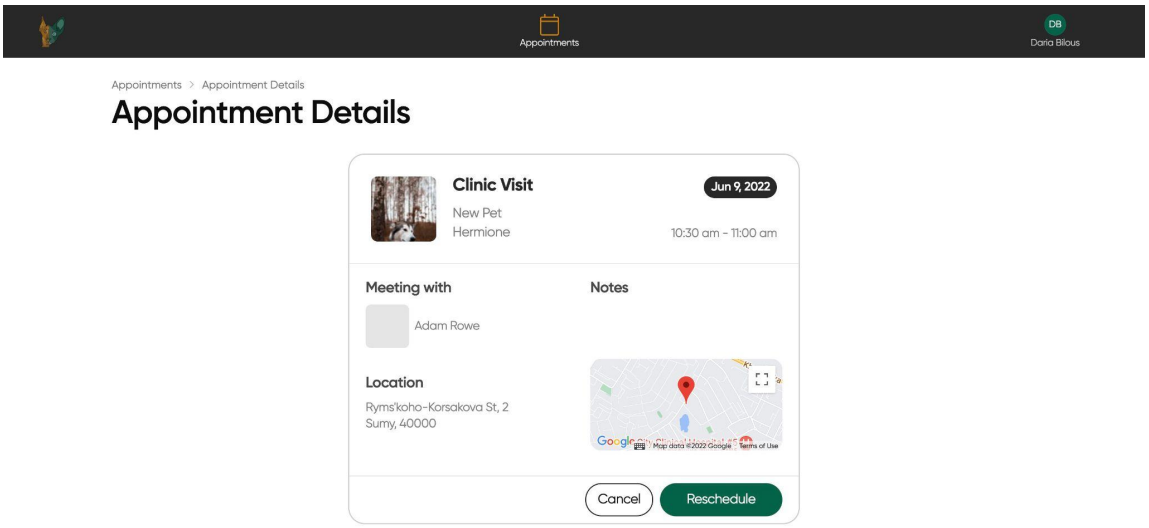


Рисунок 1.6 - дизайн PetMedico

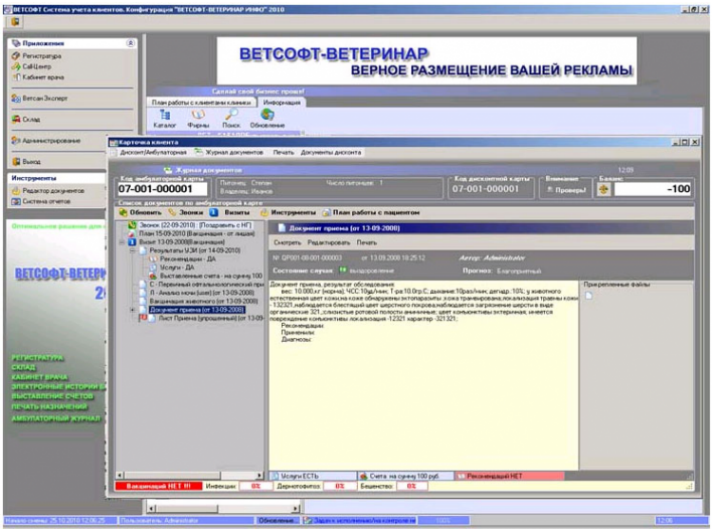


Рисунок 1.7 - дизайн ветсофт-ветеринар

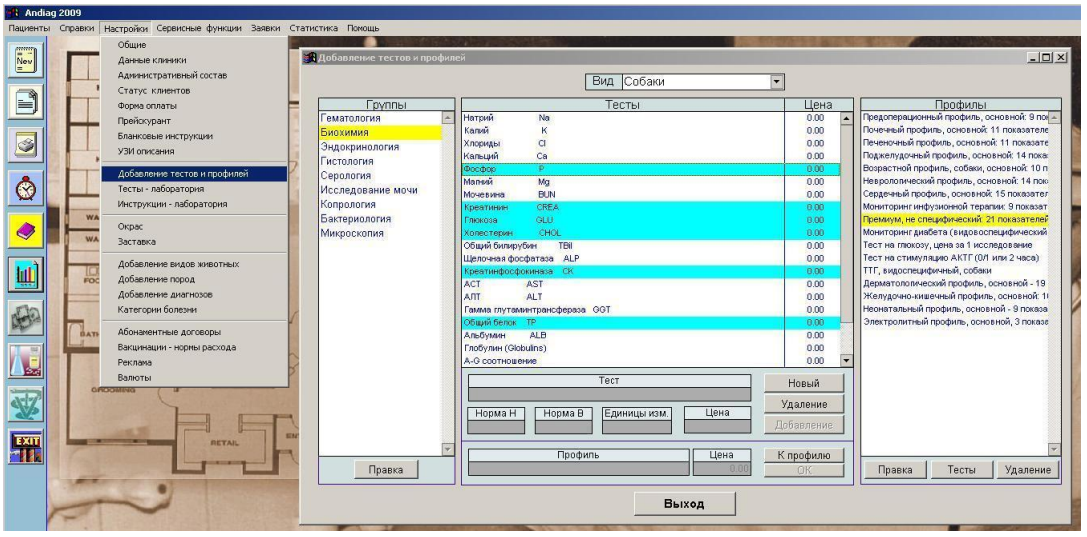


Рисунок 1.8 - дизайн андіаг

Четвертий критерій - чи пристосований функціонал додатку саме для ветеринарів. Megaplan та Appointer не є додатками лише для ветеринарів, про що є інформація на їх сайтах (Рис. 1.9 та 1.10).

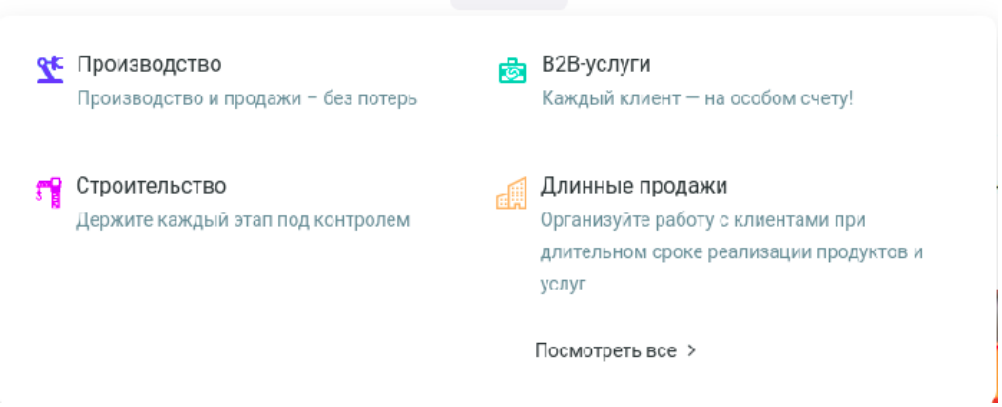


Рисунок 1.9 - сфери в яких може бути використаний додаток Megaplan

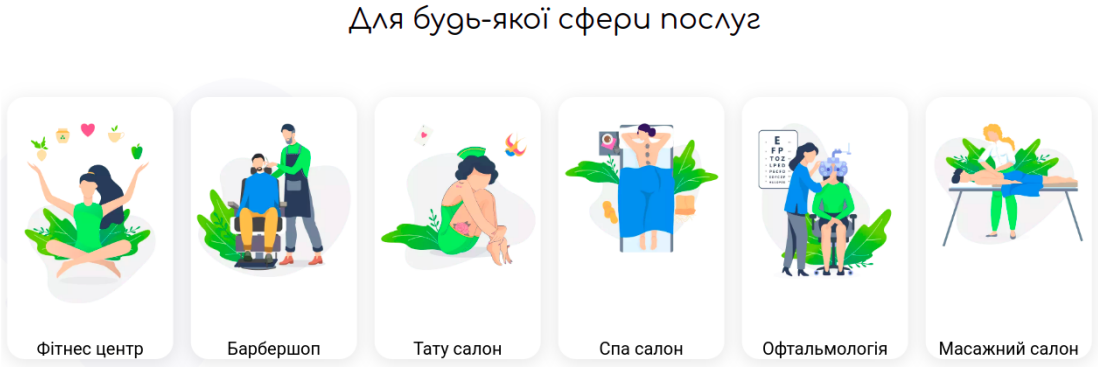


Рисунок 1.10 - сфери в яких може бути використаний додаток Appointer

Застосунки ветсофт-ветеринар та андіаг пристосовані для ветеринарних клінік, застосунок PetMedico призначений для використання лікарями ветеринарної медицини та їх клієнтами.

П’ятим критерієм є складність налаштування програмного забезпечення. Добре коли користувач може самостійно розібратися що до чого. Стосовно Megaplan є така інформація “Мегаплан пропонує гнучкі настройки, з якими зможе стартувати навіть новачок.”. PetMedico також не важкий у налаштуванні, більшість кроків при реєстрації та при подальшому користуванні застосунком достатньо інтуїтивні. На сайті застосунку Appointer є інформація що до початку користування системою, де є рекомендація “Зверніться до нас”.

Спираючись на це не можу сказати що налаштування буде легким, адже розробники припускають що клієнт самостійно не може розібратися (рис. 1.11).

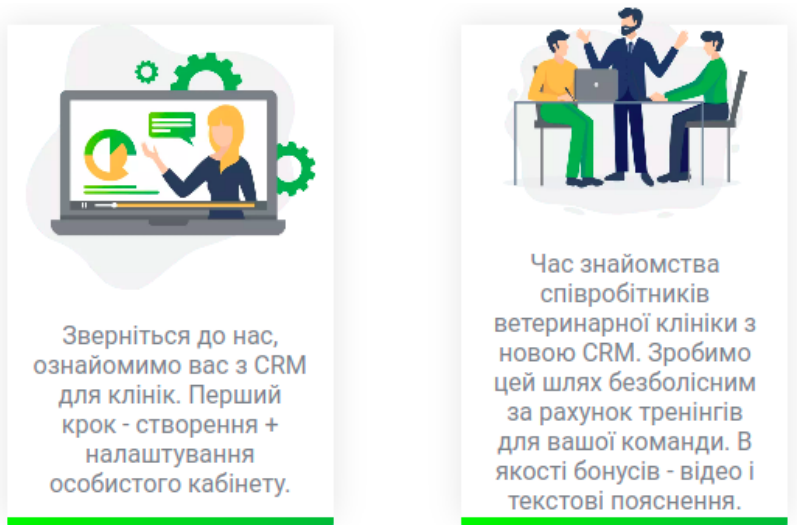


Рисунок 1.11 - початок роботи із CRM Apointer

Про програму Андіаг теж сказано - налаштування не з легких (рис. 1.12).

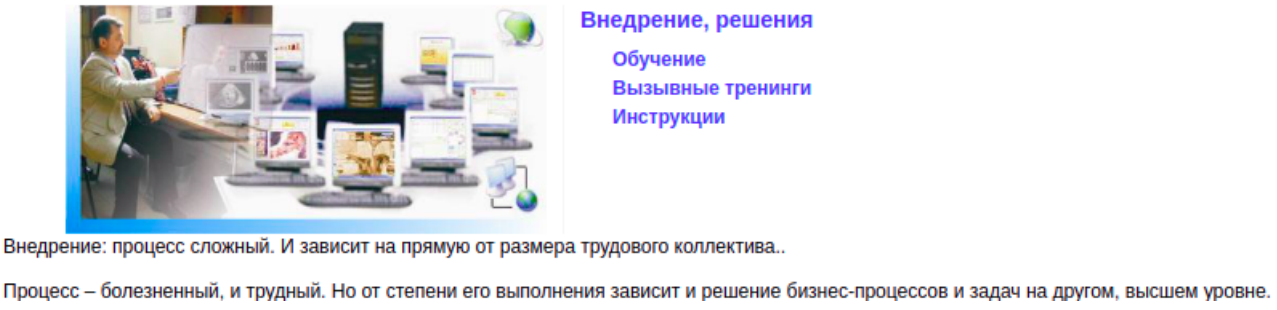
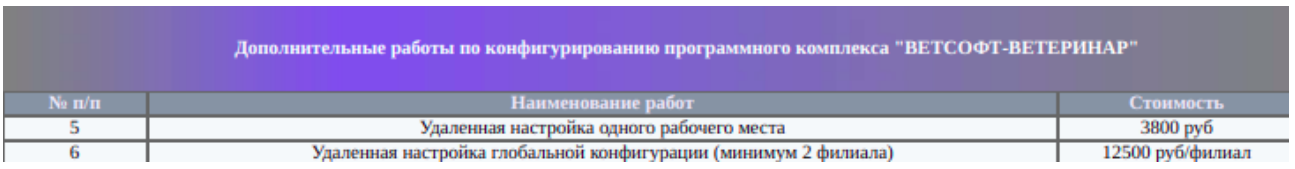


Рисунок 1.12 - початок роботи з програмою Андіаг

Для програми ветсофт-ветеринар є навіть платні послуги по налаштуванню (Рис. 1.13)



Дополнительные работы по конфигурированию программного комплекса "ВЕТСОФТ-ВЕТЕРИНАР"

№ п/п	Наименование работ	Стоимость
5	Удаленная настройка одного рабочего места	3800 руб
6	Удаленная настройка глобальной конфигурации (минимум 2 филиала)	12500 руб/филиал

Рисунок 1.13 - послуги по налаштуванню програми ветсфот-ветеринар

Наступна властивість для порівняння - чи підходить софт для приватних ветеринарів. Саме у цьому задумка PetMedico, а багато програм мають налаштування, розраховані лише на клініки. Іноді приватний ветеринар може спробувати налаштувати такі застосунки під себе, але є ймовірність що йому доведеться витрачати час на непотрібні для нього речі.

Програми ветсофт-ветеринар та андіаг відносяться до категорії “програми для ветеринарних клінік”. Не було знайдено інформації щодо використання приватними лікарями як і подібних відгуків. В Megaplan немає прикладу використання CRM приватними лікарями, є приклад для клінік (рис. 1.14), аналогічно і в CRM Appointer (рис. 1.15).

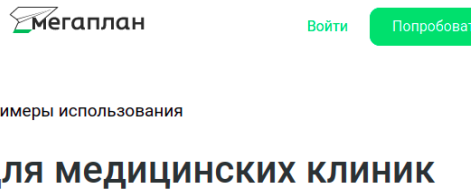


Рисунок 1.14 - приклад використання Megaplan для клінік

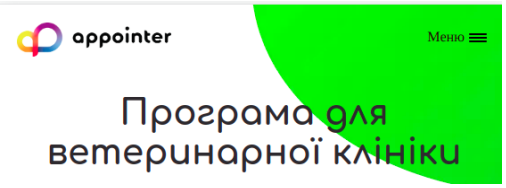


Рисунок 1.15 - приклад використання Appointer для клінік

Стосовно ролі клієнта клініки / лікаря у системі - жоден з аналогів не передбачає такої. Застосунки розраховані лише на лікарів та інших працівників клінік. Наприклад Appointer налічує такі переваги використання (Рис. 1.16):

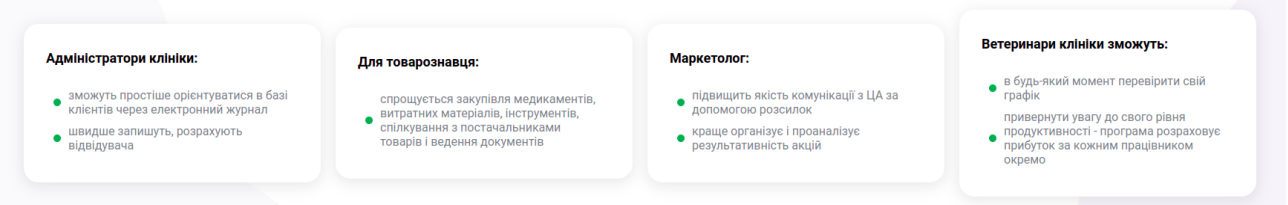


Рисунок 1.16 - на кого розрахована CRM Appointer

Останній критерій порівняння - ціна. CRM Megaplan та Appointer, програма ветсофт-ветеринар мають різні тарифи (рис. 1.17 - 1.19 відповідно). Програма Андіаг потребує лише оплату ліцензії (Рис. 1.20).

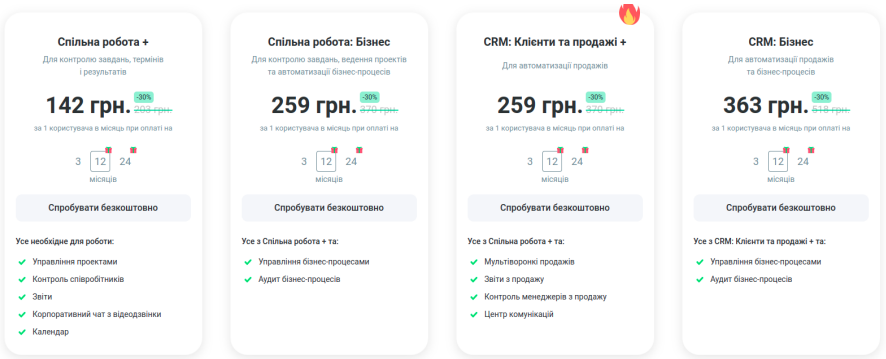


Рисунок 1.17 - ціни Megaplan

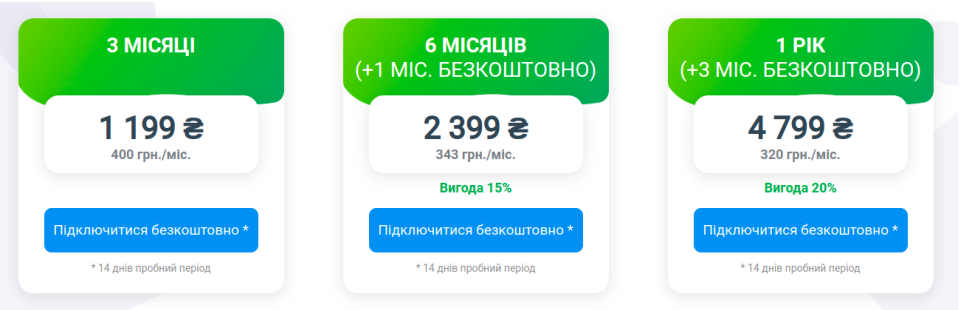


Рисунок 1.18 - ціни Appointer

Программный комплекс ВЕТСОФТ ВЕТЕРИНАР - НОРМАЛ	Локальная	17900
Программный комплекс ВЕТСОФТ ВЕТЕРИНАР - МАЛЫЙ БИЗНЕС	Сетевая, до 3-х подключений	25700
Программный комплекс ВЕТСОФТ ВЕТЕРИНАР - УСПЕШНЫЙ	Сетевая, до 5-и подключений	40700
Программный комплекс ВЕТСОФТ ВЕТЕРИНАР - СУПЕР	Сетевая, до 10-и подключений	75900
Программный комплекс ВЕТСОФТ ВЕТЕРИНАР - МАГНАТ	Сетевая, без учета числа подключений	договорная

Рисунок 1.19 - ціни ветсофт-ветеринар

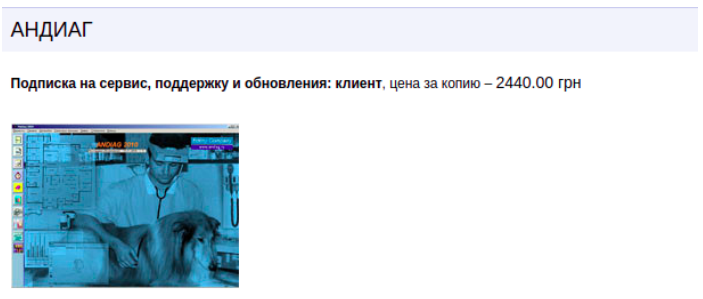


Рисунок 1.20 - ціна Андіаг

Детальне порівняння завершено, коротко результати можна переглянути у таблиці 1.1.

Таблиця 1.1 - порівняння з аналогами

Аналоги / Критерії порівняння	**PetMedico**	Megaplan	Ветсофт - ветеринар	Андіаг	Appointer
Не потребує установки	+	+	-	-	+
Має мобільну версію	+	+	-	-	+
Сучасний дизайн	+	+	-	-	+
Функціонал пристосований до ветеринарної медицини	+	-	+	+	-
Не складний у налаштуванні	+	+	-	-	-
Підходить для приватних ветеринарів (не має багато зайвого функціоналу)	+	-	-	-	-
Має інтерфейс для клієнта	+	-	-	-	-
Безкоштовний	+	-	-	-	-

1.3 Постановка задачі дослідження

Метою даного дипломного проекту є автоматизація процесів у взаємодії власників тварин із лікарями ветеринарної медицини шляхом створення інформаційної системи, що призведе до полегшення та збільшення ефективності роботи ветеринарів, котрі працюють самостійно, а також буде економити час людей, яким потрібні послуги ветеринара. Система дасть змогу таким лікарям заявити про себе навіть не маючи власного сайту, а для клієнтів зможе розширити лист спеціалістів, до яких вони можуть звернутися, та дасть змогу переглядати інформацію про свою тварину без звернення у клініку. У системі розрізняються дві основні ролі - клієнт і лікар ветеринар.

Функціональні вимоги системи для клієнта:

- авторизація у системі
- самостійна реєстрація, при якій обов’язково треба вказати одну домашню тварину
- можливість додавати домашніх тварин
- можливість бронювати запис онлайн та керувати їм
- можливість переглядати та редагувати власний профіль
- можливість переглядати та редагувати профіль тварини
- можливість переглядати історію прийомів та рекомендації лікаря по них, показники здоров’я тварини

Функціональні вимоги системи для ветеринара:

- можливість переглядати та редагувати власний профіль
- можливість залишати призначення після консультації
- можливість переглядати профіль тварини для отримання необхідної для консультації інформації

2 МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ

2.1 Методи реалізації

Для розробки інформаційної системи було обрано такі засоби реалізації:

- TypeScript - строго типізована мова програмування, яка базується на JavaScript. У даному проекті використовується на фронтенді. Також на фронтенді використовується React. Це JavaScript бібліотека для створення інтерфейсів користувача. Багато JS бібліотек вміють працювати і з TS.
- Ruby - динамічна мова програмування з відкритим вихідним кодом з акцентом на простоту та продуктивність. Мова має простий синтаксис, який природний для читання та легкого розуміння. У даному проекті Ruby використовується на бекенді. Бекенд відповідає правилам RESTful API. REST розшифровується як репрезентаційна передача стану. REST API засновані на уніфікованому ідентифікаторі ресурсу і протоколі HTTP, можуть обмінюватися даними в форматі JSON або XML.
- PostgreSQL - потужна система управління реляційними базами даних з відкритим вихідним кодом з більш ніж 30-річною активною розробкою, завдяки якій вона заслужила міцну репутацію за надійність і продуктивність. Спільнота розробників досить швидко реагує на репорти багів, а офіційна документація добре структурована та зрозуміла. Завдяки цьому використовувати PostgreSQL досить просто, і для багатьох проблем вже є готові рішення.
- OnSched - платформа для планування подій, працювати з якою можна за допомогою REST API та яка має вичерпну документацією. OnSched створює індивідуальні рішення для бронювання для потреб відділів продажів B2B, каталогів, маркетплейсів та іншого. Інтеграція з даною платформою реалізована на апі. Можливості платформи використовуються для створення записів до лікарів.

2.2 Побудова діаграм за методологіями IDEF та DFD

Основні користувачі системи – лікарі ветеринарної медицини та їх клієнти. За допомогою розробляємої інформаційної системи клієнти зможуть знайти потрібного лікаря згідно проблеми своєї тварини та записатися до нього на прийом онлайн. Ветеринари зможуть переглядати розклад прийомів, залишати рекомендації для власника тварини, що б той пізніше міг їх переглядати.

Крім ветеринарів та клієнтів є третя особа, яка не приймає участь безпосередньо у функціонуванні системи. Ця особи повинна перевіряти людину, що хоче зареєструватися як ветеринар. Після заповнення кандидатом форми запиту до реєстрації, з ним зв’язуються поштою, він надає документи, що підтверджують його право займатися ветеринарною практикою (ліцензія, диплом, сертифікати про закінчення курсів). Документи перевіряються, якщо все гаразд, кандидат отримує лист з логіном і паролем, який може змінити потім. Процеси системи відображені на IDEF та DFD діаграмах (Рис 2.1 і 2.2).

Рисунок 2.1 - IDEF0 інформаційної системи

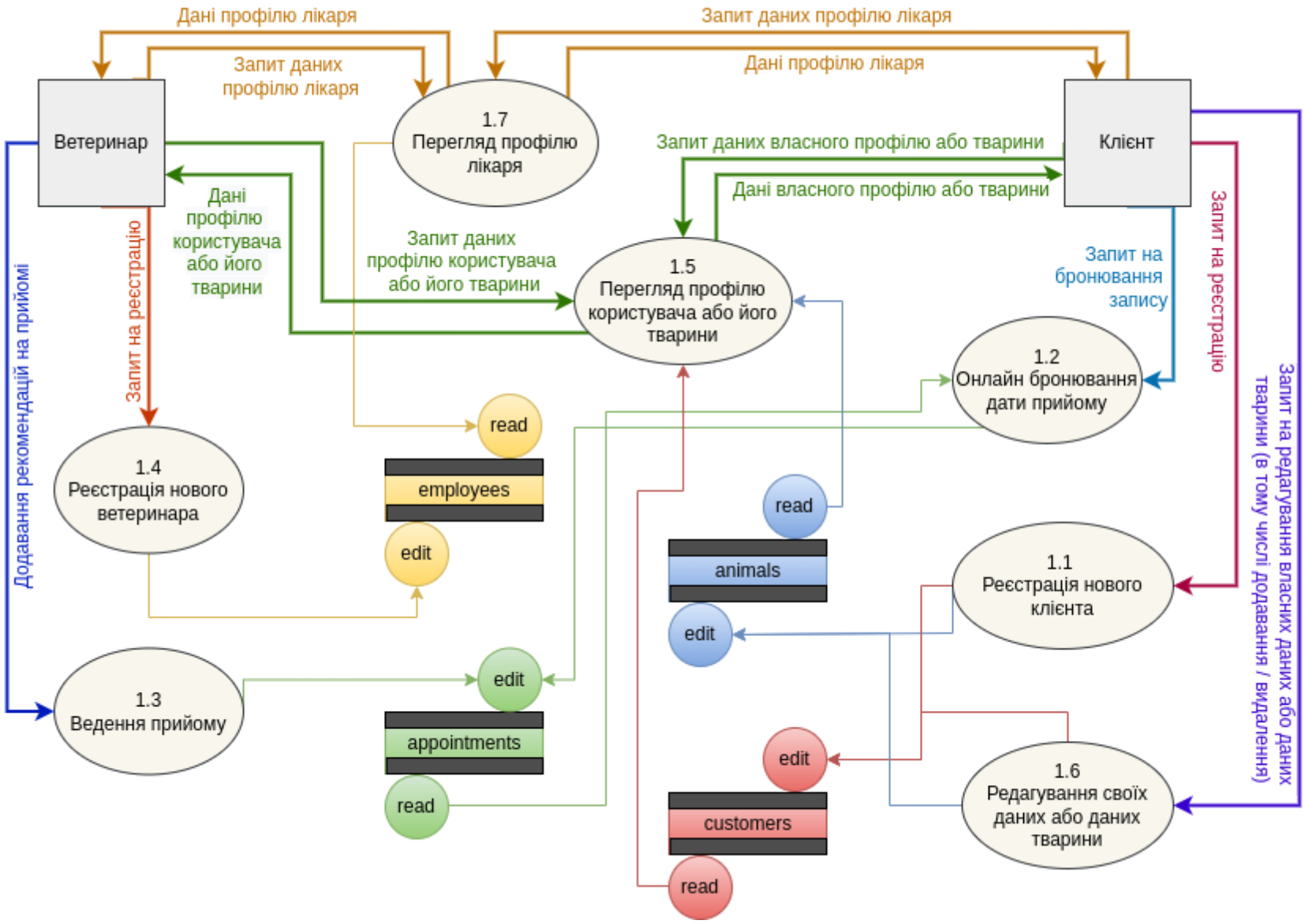


Рисунок 2.2 - DFD level 1 інформаційної системи

Для обробки процесів концептуальна модель на рис. 2.2 має наступні сховища даних: «Employees» (зберігаються лікарі ветеринари), «Customers» (зберігаються клієнти), «Animals» (зберігаються домашні тварини клієнтів, яких вони додали до системи самостійно), «Appointments» (зберігаються записи до лікарів, як майбутні, так і минулі). Крім цих чотирьох таблиць в базі є й інші, але їх не внесено до моделі, тому що вони не основні. Деякі з них допоміжні для інтеграції з onSched, в деяких зберігається додаткова інформація про лікарів та візити.

Клієнт має доступ до трьох процесів. «Реєстрація нового клієнта» - тут клієнт заповнює всі необхідні дані про себе та додає одну тварину (через це на схемі процес взаємодіє з двома сховищами). «Онлайн бронювання дати прийому» - тут клієнт обирає причину звернення до лікаря, потім самого ветеринара і час, який його влаштовує для візиту. «Редагування своїх даних або даних тварини» - у системі клієнт має можливість додавати нових тварин, видаляти існуючих, та редагувати дані у власному профілі.

Два процеси доступні як клієнту, так і лікарю, але з різними обмеженнями. «Перегляд профілю лікаря» - клієнт має можливість переглянути профіль обраного лікаря, лікар має можливість переглядати власний профіль. «Перегляд профілю користувача або його тварини» - ветеринар та сам користувач можуть переглядати дані профілю, а також дані тварини, де можуть знаходитись історія візитів до лікаря та рекомендації.

Ветеринар має доступ ще до двох процесів. «Реєстрація нового ветеринару» - людина має можливість заповнити форму для запиту реєстрації як ветеринар. «Ведення прийому» - лікар має можливість записувати рекомендації для тварини під час візиту.

Звернення до баз даних розділені на дві основні групи - запити на читання та запити на редагування даних. До редагування відносяться зміна, видалення та додавання. Що б схема було легко читати запити не супроводжуються описом, адже записи одного типу з різних процесів схожі.

2.3 Побудова діаграми прецедентів

Діаграма прецедентів візуально зображає різноманітні сценарії взаємодії між акторами (користувачами) і прецедентами (випадками використання). Нижче наведено інформацію щодо акторів (табл. 2.1) та опис варіантів використання (табл. 2.2) для розробляємої інформаційної системи.

Табл. 2.1 - Опис акторів

Актор	**Опис**
Клієнт	Власник тварини, який хоче звернутися до лікаря
Ветеринар	Лікар ветеринарної медицини, який надає послуги

Табл. 2.2 - Опис варіантів використання

Варіант використання	**Опис**
Реєстрація	Можливість реєстрації до системи
Авторизація	Можливість входу до системи
Перегляд та редагування даних власного профілю	Можливість переглядати. та вносити зміни у дані власного профілю користувача
Перегляд запланованих та проведених візитів	Можливість переглядати проведені та заплановані візити тварини клієнта, та рекомендації по проведених візитах
Запис на прийом	Можливість онлайн керувати своїм записом
Додавання тварини	Можливість додати нову домашню тварину до профілю
Редагування даних тварини	Можливість змінювати інформацію про тварину
Додавання рекомендацій	Можливість записувати рекомендації протягом візиту
Перегляд даних тварин клієнтів	Можливість переглядати дані тварин клієнтів

Сама діаграма використання наведена на рисунку 2.3

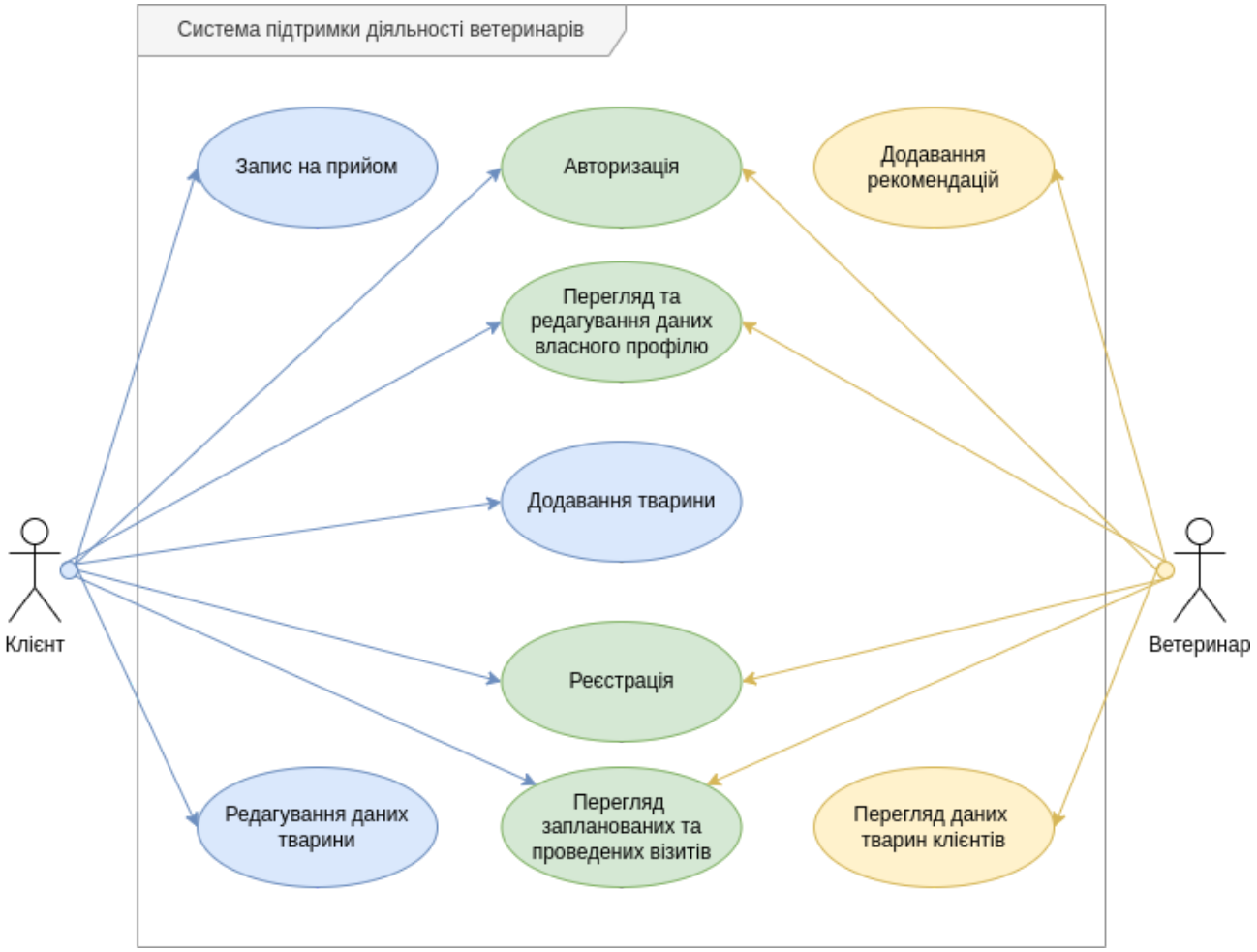


Рисунок 2.3 - Діаграма варіантів використання

2.4 Створення ескізів додатку

Для даної інформаційної системи не створювався деталізований дизайн, а лише мокапи, щоб приблизно розуміти на яких сторінках які елементи будуть відображатися. Мокап - це візуальний прототип, який показує основну структуру сторінки та основні UI елементів на ній. Такий прототип це лише начерк, основна ідея, тому часто його роблять у мальованому стилі. Створювати мокапи зручно коли потрібно швидко промоделювати новий продукт. На цьому етапі не мають значення основні кольори та зовнішній вигляд елементів, лише їх розміщення.

Почнемо зі сторінки реєстрації. Ліва частина екрана буде зайнята картинкою, пов’язаною з тваринами, а в правій частині буде форма для реєстрації та посилання на форму для авторизації, якщо користувач вже має акаунт. Також в лівому верхньому куті буде логотип інформаційної системи, а над формою авторизаці її назва. Все це зображено на рис. 2.4.

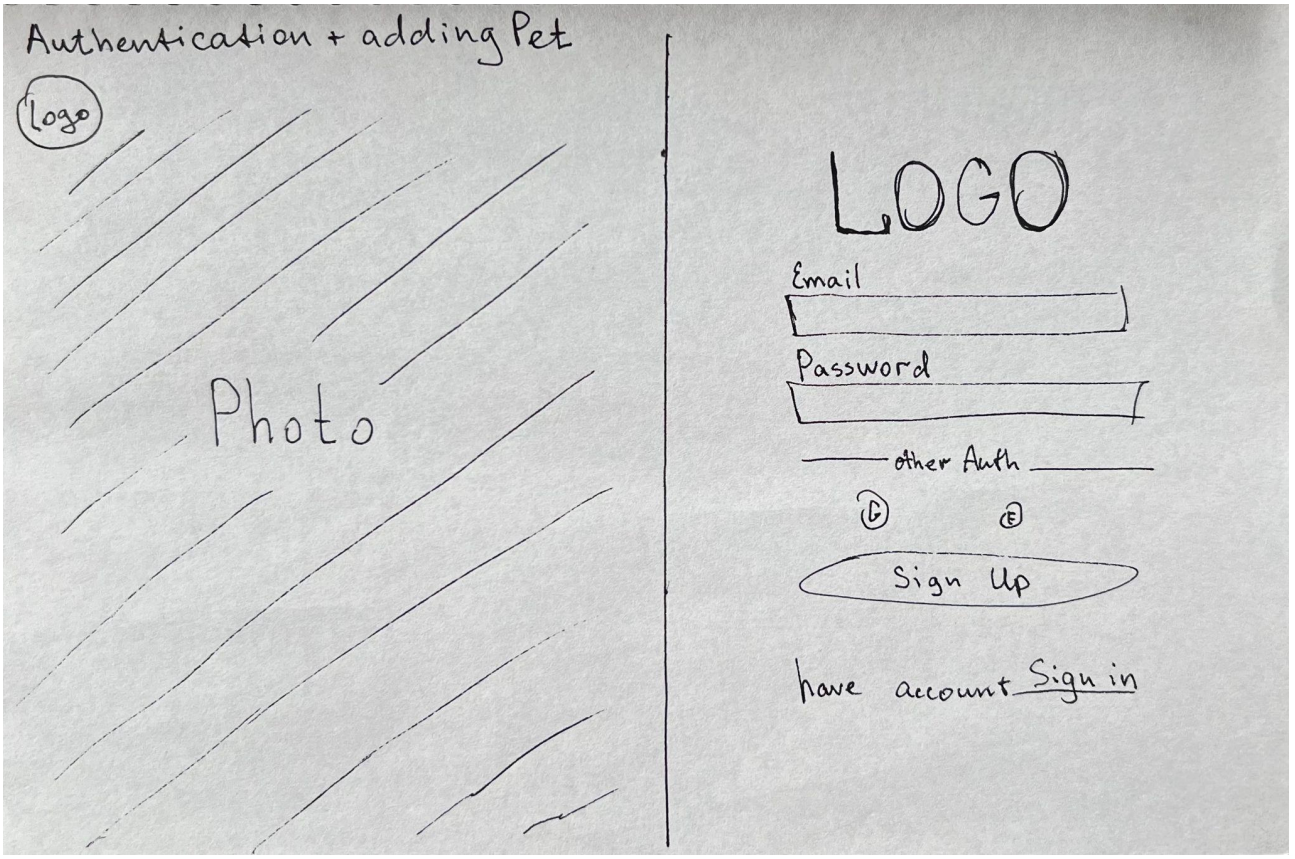


Рисунок 2.4 - мокап сторінки авторизації

Якщо користувач ще не має акаунту, то після введення поштової скриньки та паролю він потрапить на сторінку введення даних тварини для продовження реєстрації у системі. Для того щоб зареєструватися потрібно мати хоча б одну тварину, тому її додавання обов’язкове під час реєстрації. Процес заповнення даних про тварину має декілька кроків, усі відбуваються на одній сторінці, змінюється лише номер кроку та блок з питанням. У блоку питань буде футер, у якому можуть розміщуватися посилання на додаткову для клієнта інформацію, яка допоможе йому дати відповідь, та кнопка для підтвердження відповіді.

Сторінка зображена на рис. 2.5, блок питань на різних кроках - рис. 2.6.

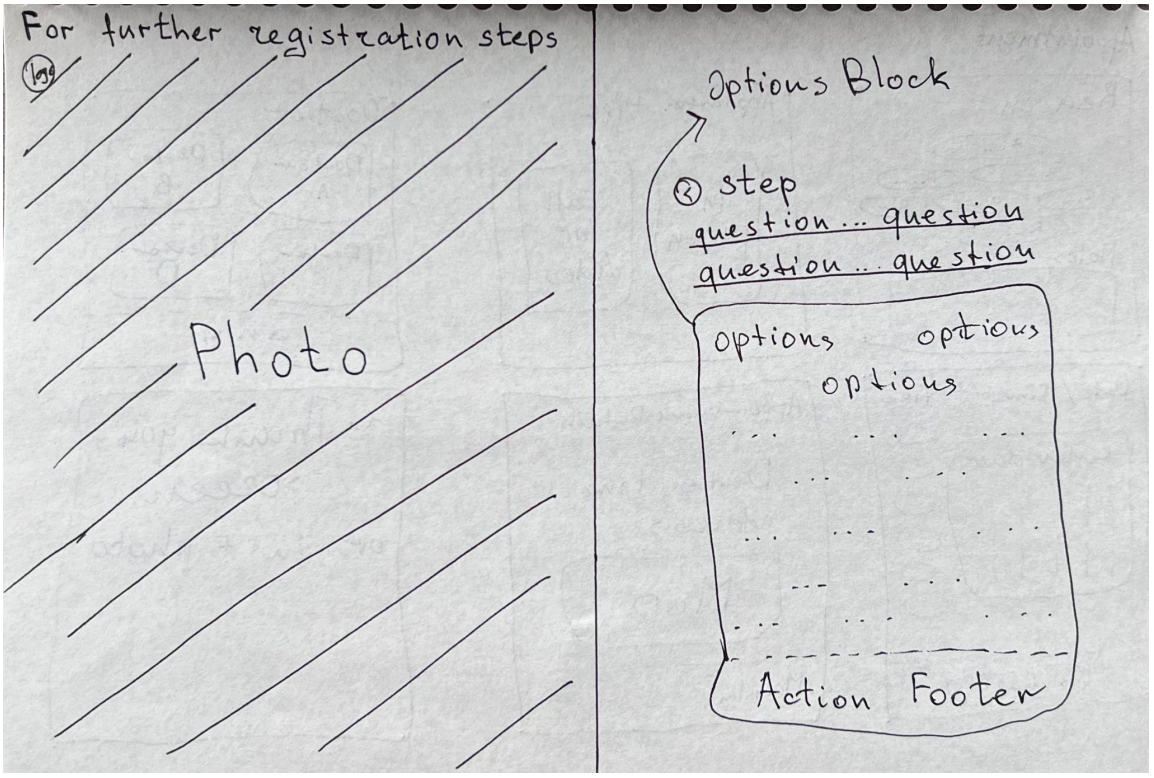


Рисунок 2.5 - мокап сторінки додавання тварини під час реєстрації

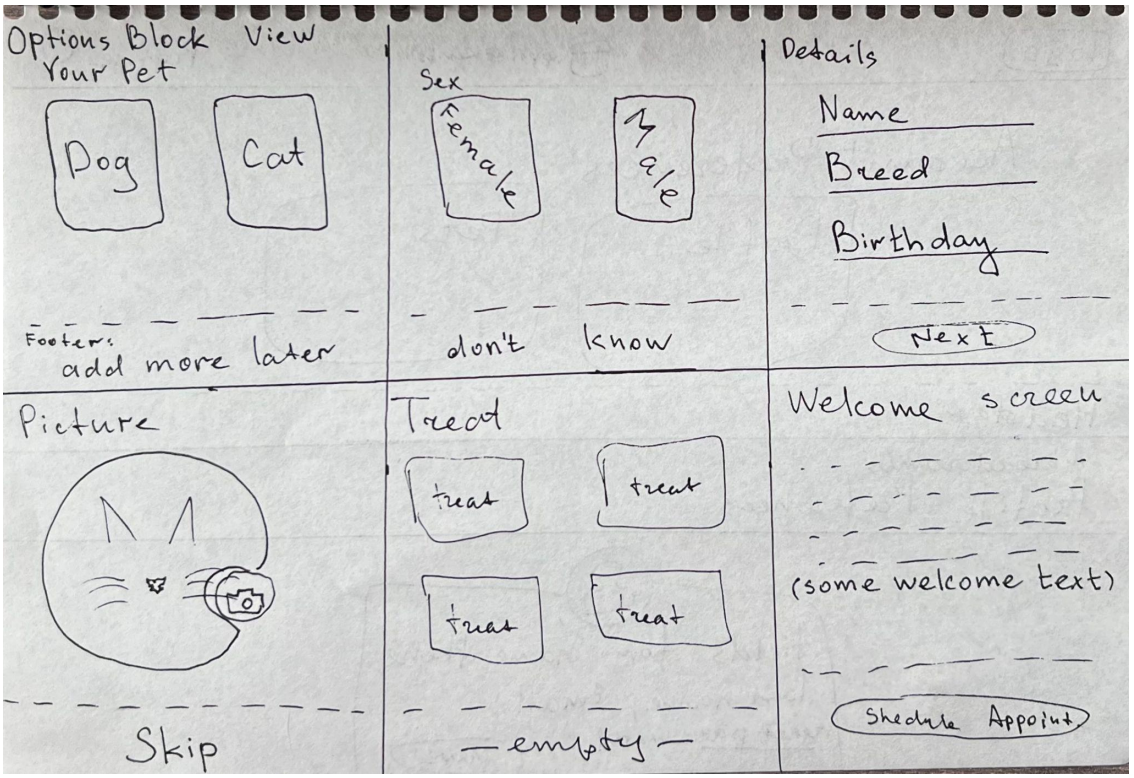


Рисунок 2.6 - мокап блоку запитань на різних кроках

Сторінка створення запису до лікаря матиме схожий вид із сторінкою додавання тварини, лише вид блоку питань буде відрізнятися. Для створення запису блок питань зображено на рис. 2.7

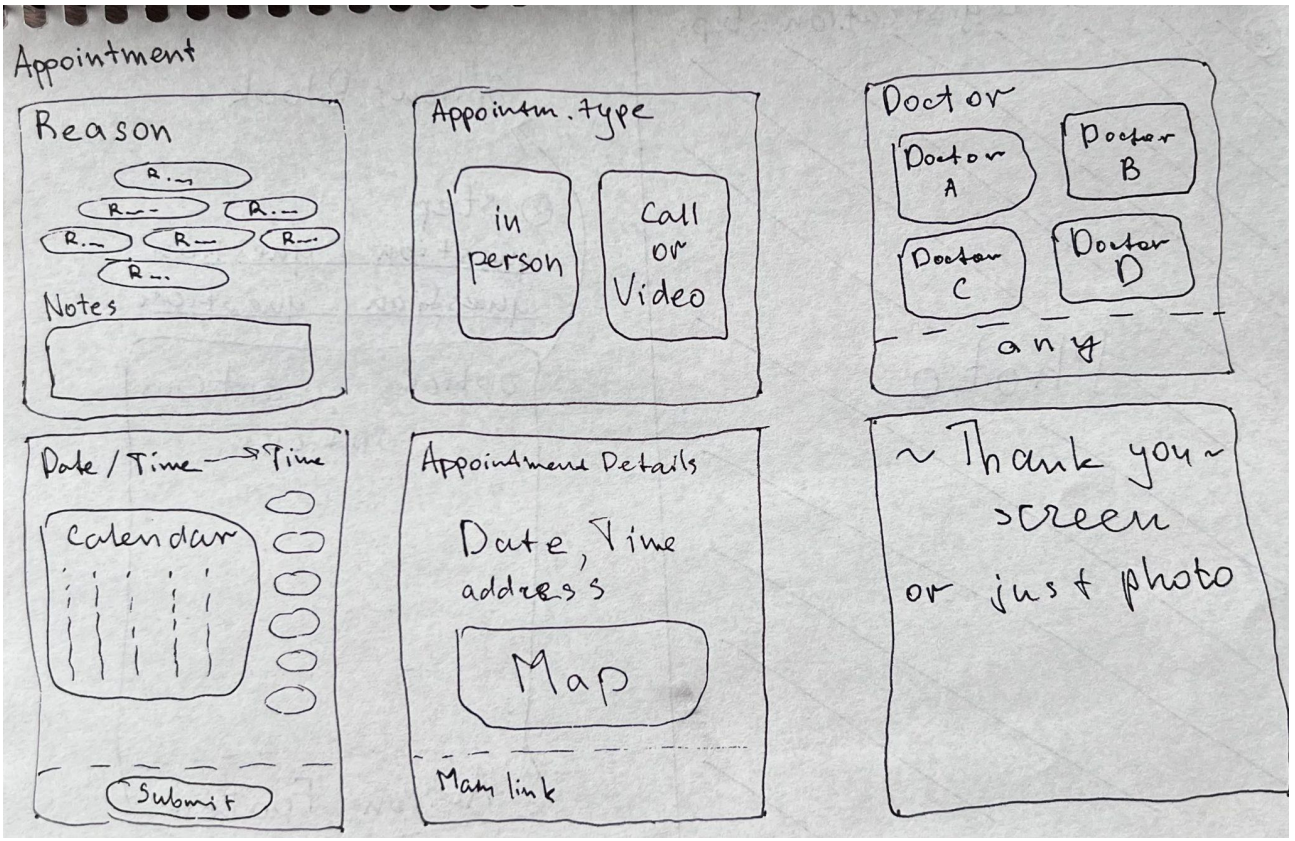


Рисунок 2.7 - мокап блоку питань для створення запису до лікаря

При переході до акаунту користувач буде бачити сторінку, що зображена на рис. 2.8. З цієї сторінки він зможе перейти до сторінки своєї тварини або свого власного профілю.

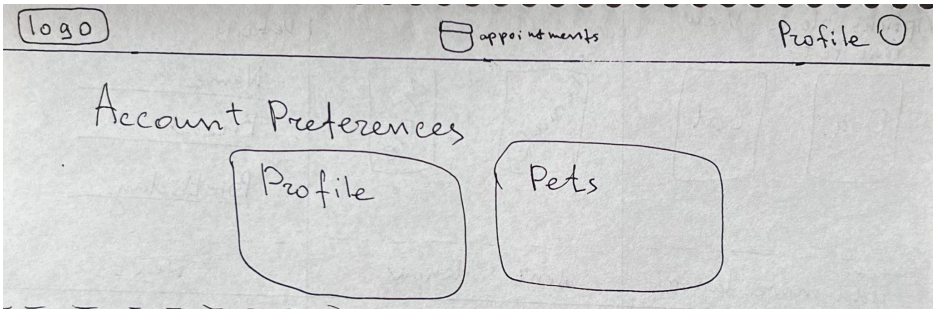


Рисунок 2.8 - мокап сторінки користувача

При переході до власного профілю користувачу буде відображатись форма із його фото зверху, та полями, які можна редагувати (не всі поля будуть доступні для змін). Рис. - 2.9. На мокапі відображено не всі поля та їх розміщення, тому що на даному етапі не відомо повний набір таків полів.

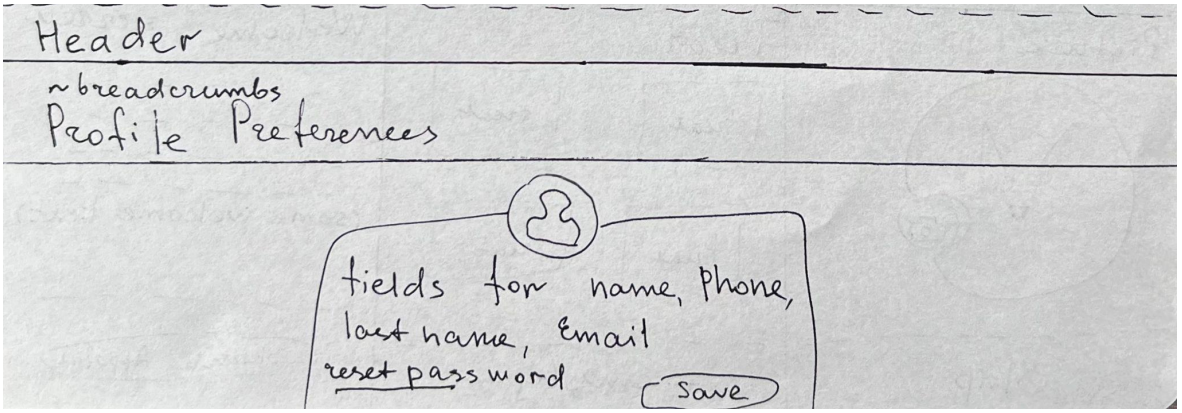


Рисунок 2.9 - мокап сторінки із формою профілю користувача

Профіль тварини буде схожий на профіль користувача, але з полями під дані тварини відповідно. Можна змінювати фото, ім’я, додавати інформацію щодо характеру та особливостей тварини. Профіль тварини - рис. 2.10

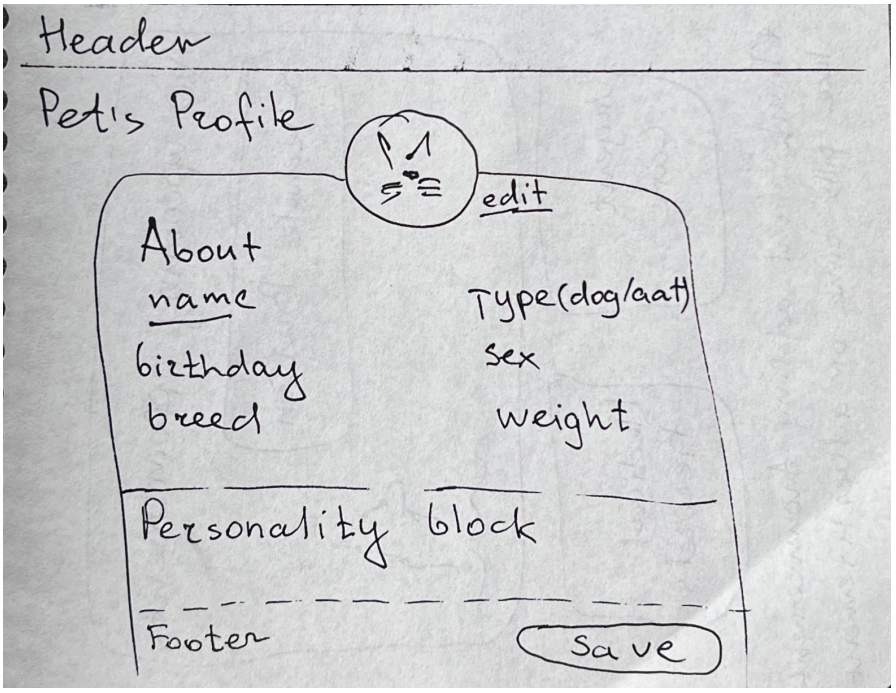


Рисунок 2.10 - мокап профілю тварини

Система матиме дашборд (головна сторінка), на якому буде зібрано важливу для користувача інформацію. Вид дашборда може трішки відрізнятись в залежності від того чи є записи та якісь документи. Обидва види зображено на рис. 2.11.

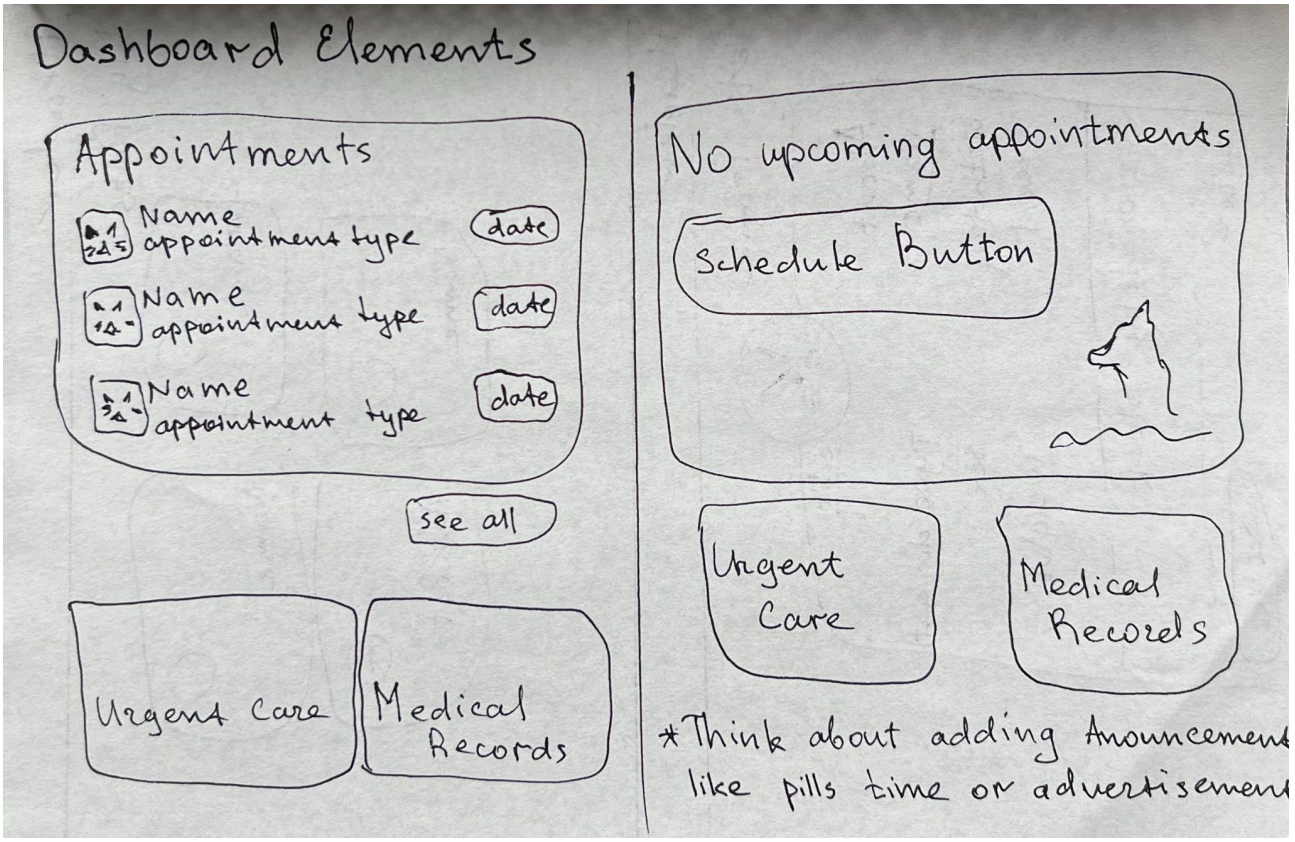


Рисунок 2.11 - мокап дашборду

Кожна сторінка матиме шапку, у якій буде іконка профілю користувача, меню зі сторінками системи, логотип та назва. Шапку промальовано не на кожному мокапі, тому що її вигляд та положення не змінюється. Лише може бути виділений активний пункт меню, відповідно до сторінки на який знаходиться користувач.

У системі буде сторінка записів “Appointments”. Записи можна буде переглядати для всіх тварин одразу, або для кожної окремо. З цієї сторінки буде можливість перейти до створення нового запису, відміни чи корегування існуючого, та буде можливість переглядати історію попередніх записів. Форма

для редагування запису буде містити календар та чіпи з доступним часом. Ця частина системи зображена на рис. 2.12.

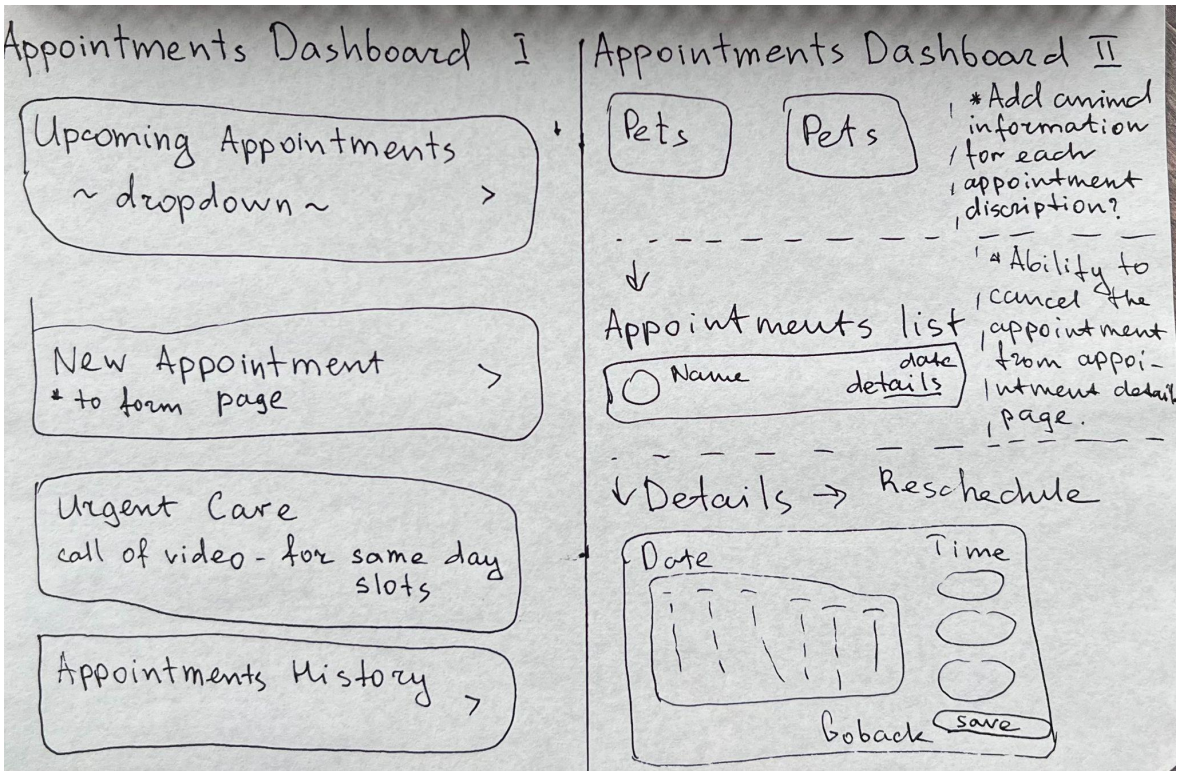


Рисунок 2.12 - мокап сторінки записів

Сторінка зі списком тварин доволі проста, кожній тварині відповідає блок, у якому є її ім’я та фото. Останній блок у списку для додавання нової тварини. Список тварин зображено на рис. 2.13

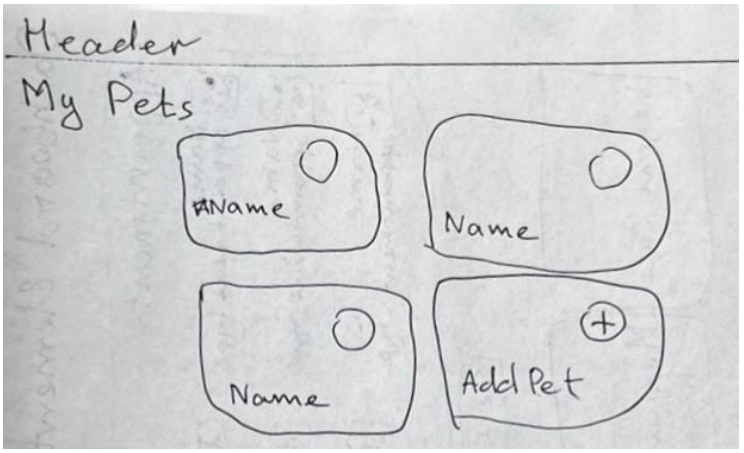


Рисунок 2.13 - мокап списку тварин

В системі будуть чотири кольори. Сірий та білий - для великих елементів, зелений та помаранчевий для акцентів (кнопки, іконки, назви, посилання, підкреслення і тд.).

- темно-сірий - *393939ff*
- білий - *#ffffffff*
- зелений - *#006349ff*
- помаранчевий - *#c87e00ff*

Логотип для інформаційної системи було створено у програмі Adobe Illustrator. Для зображення були використані два акцентні кольори - зелений і помаранчевий. Вид логотипу приведено на рис. 2.14.

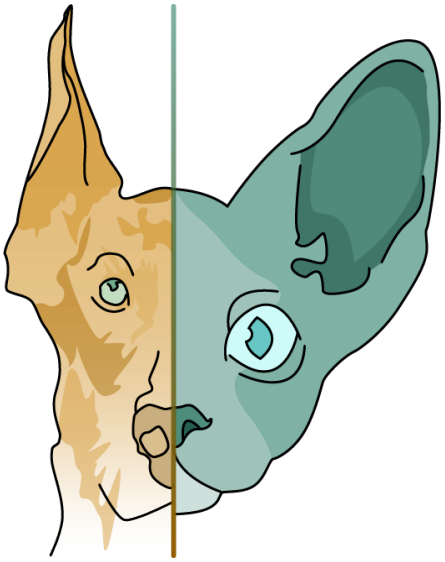


Рисунок 2.14 - логотип інформаційної системи

Назва системи PetMedico була зроблена у форматі svg для зручності використання. Повна та скорочена назви наведено на рис. 2.15 і 2.16.

PetMedico

Рисунок 2.15 - повна назва системи

PM

Рисунок 2.16 - скорочена назва системи

2.5 Опис бази даних

Логічна модель бази даних допоможе отримати графічне представлення структури таблиць та зв’язків між ними. Створена база даних для розробляємої інформаційної системи містить дванадцять таблиць. Їх перелік наведено нижче:

- Customers (користувачі) - зберігається інформація про клієнтів, має зовнішній ключ на таблицю з адресами
- Addresses (адреси) - зберігаються адреси, на котрі потім посилаються таблиці користувачів та ветеринарів
- Animals (тварини) - зберігаються дані про тварин клієнтів
- Species (види) - зберігаються доступні види тварин
- Breeds (породи) - зберігаються породи тварин
- Employees (співробітники - ветеринари) - зберігається інформація про ветеринарів, має зовнішній ключ на таблицю з адресами
- Visit_reasons (причини візитів) - зберігаються доступні причини візиту
- Visit_reason_employee (причини візитів - ветеринар) - проміжна таблиця для зв’язку між visit_reasons та employees. Зберігається інформація який ветеринар з якими причинами візиту працює
- Appointments (записи) - зберігаються записи клієнтів до ветеринарів
- Consults (консультації) - зберігаються дані по консультації (відбувшийся запис)
- Prescription (призначення) - зберігаються призначення ветеринара по лікуванню тварини
- Health_statuses (показники здоров’я) - зберігаються різні показники тварини (наприклад тиск, вага, рівень болю, результати аналізів). Лікар має можливість записувати ці дані на кожній консультації

Модель бази даних наведено на рисунку 2.17

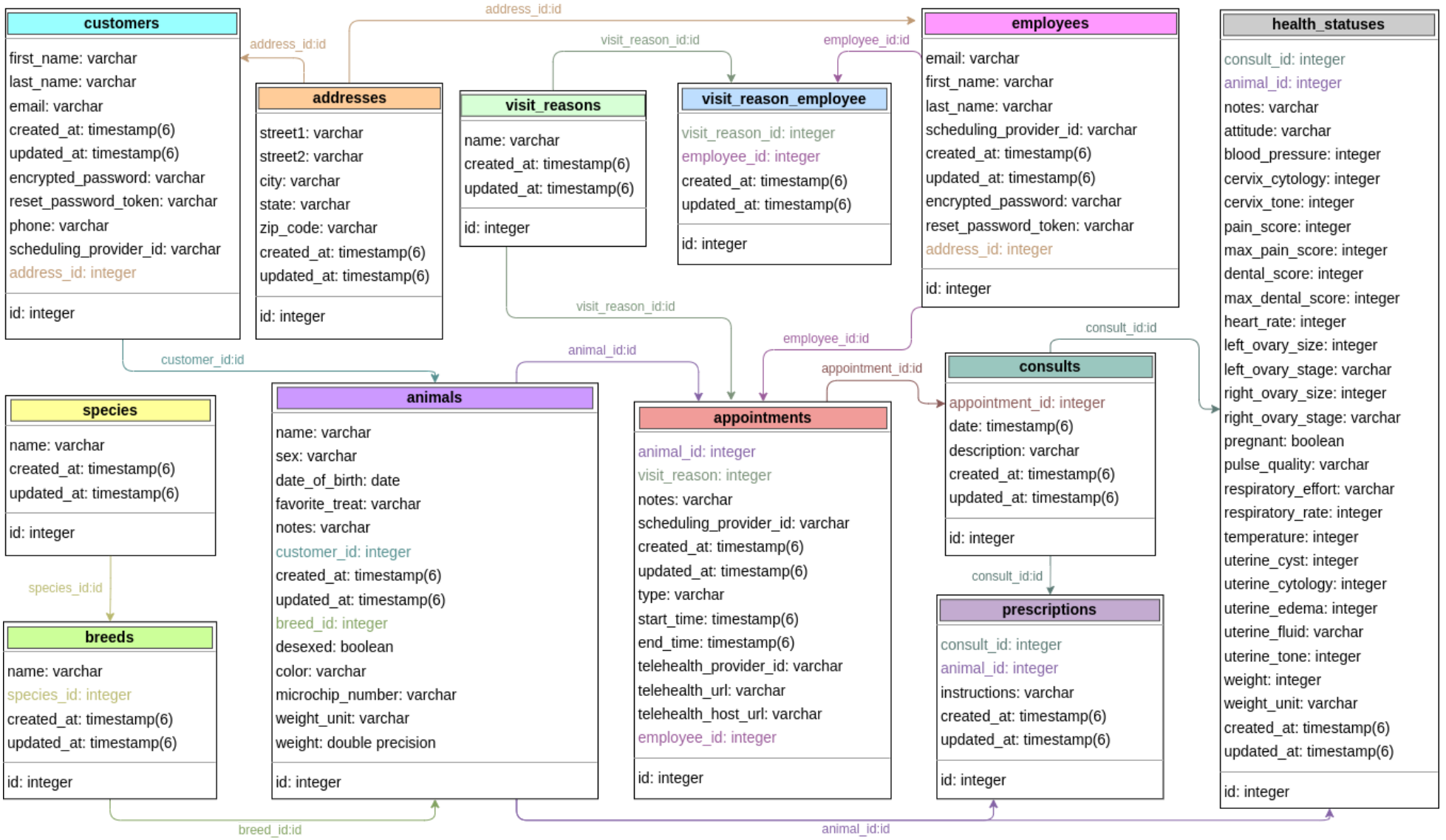


Рисунок 2.17 - модель бази даних

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

Перейшовши до сайту системи користувач потрапить на сторінку реєстрації / авторизації. Сторінка містить поля “email” та “password”, кнопку “Sign Up” / “Sign In” та посилання для переходу між реєстрацією та авторизацією. Сторінку зображено на рисунку 3.1.

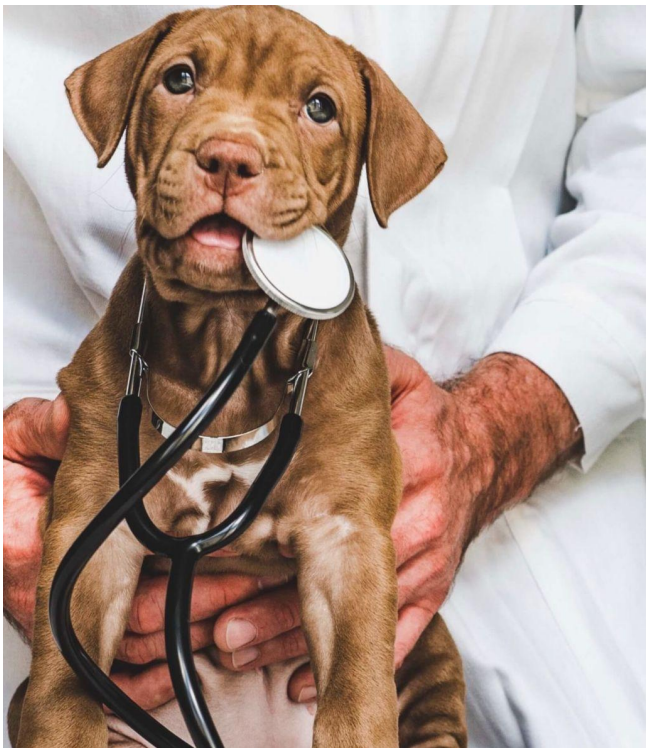


Рисунок 3.1 - сторінка реєстрації / авторизації

Якщо користувач обирає реєстрацію, то після введення електронної пошти і паролю він потрапить на сторінку із заповненням даних про свою тварину, яка матиме п’ять кроків для її додавання. Якщо ж користувач вже має акаунт у системі, то після введення коректних електронної пошти і паролю він потрапить на основний дашборд системи. Вид даної сторінки буде наведено згодом.

Перший крок при додаванні тварини під час реєстрації пропонує обрати тип тварини, власником якої є користувач. На сторінці є степер, блок з питанням та варіантами відповідей. Зображення сторінки - рисунок 3.2.

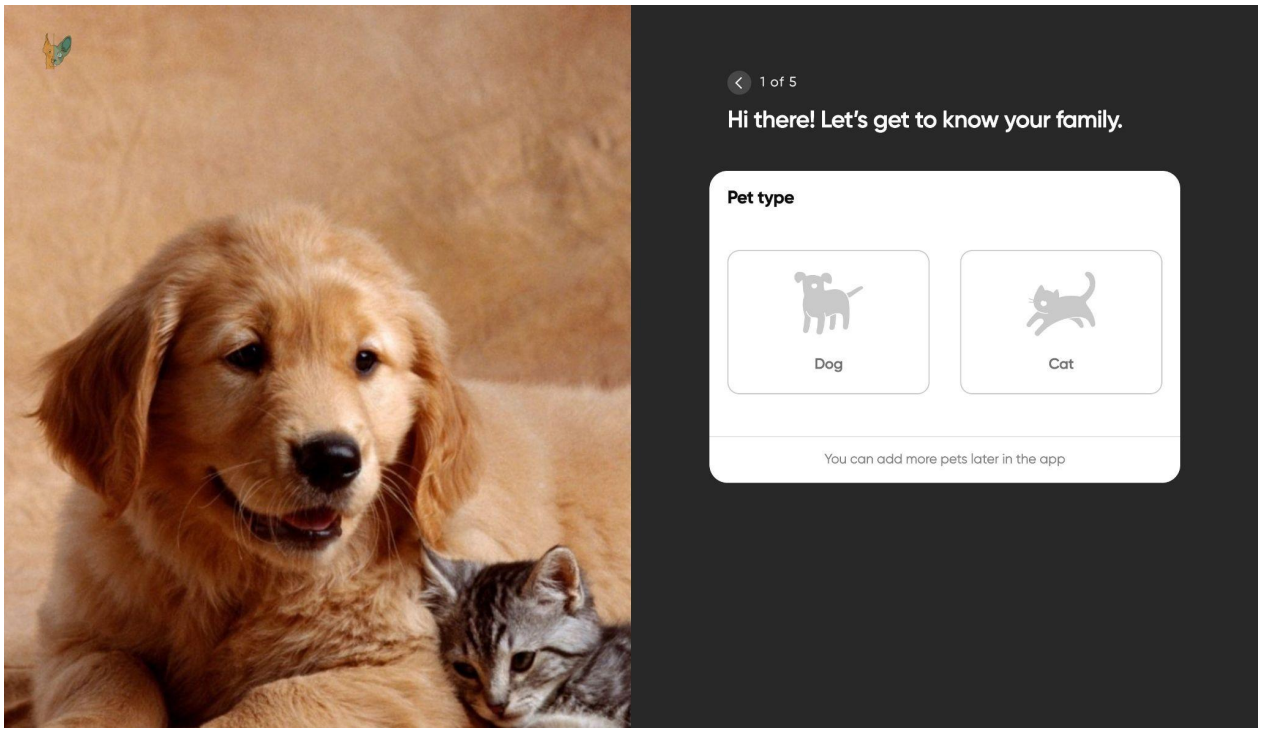


Рисунок 3.2 - перший крок додавання тварини

На другому кроці потрібно обрати пол тварини, доступний варіант “I’m not sure” для тих людей, які ще не знают його. Крок зображено на рисунку 3.3.

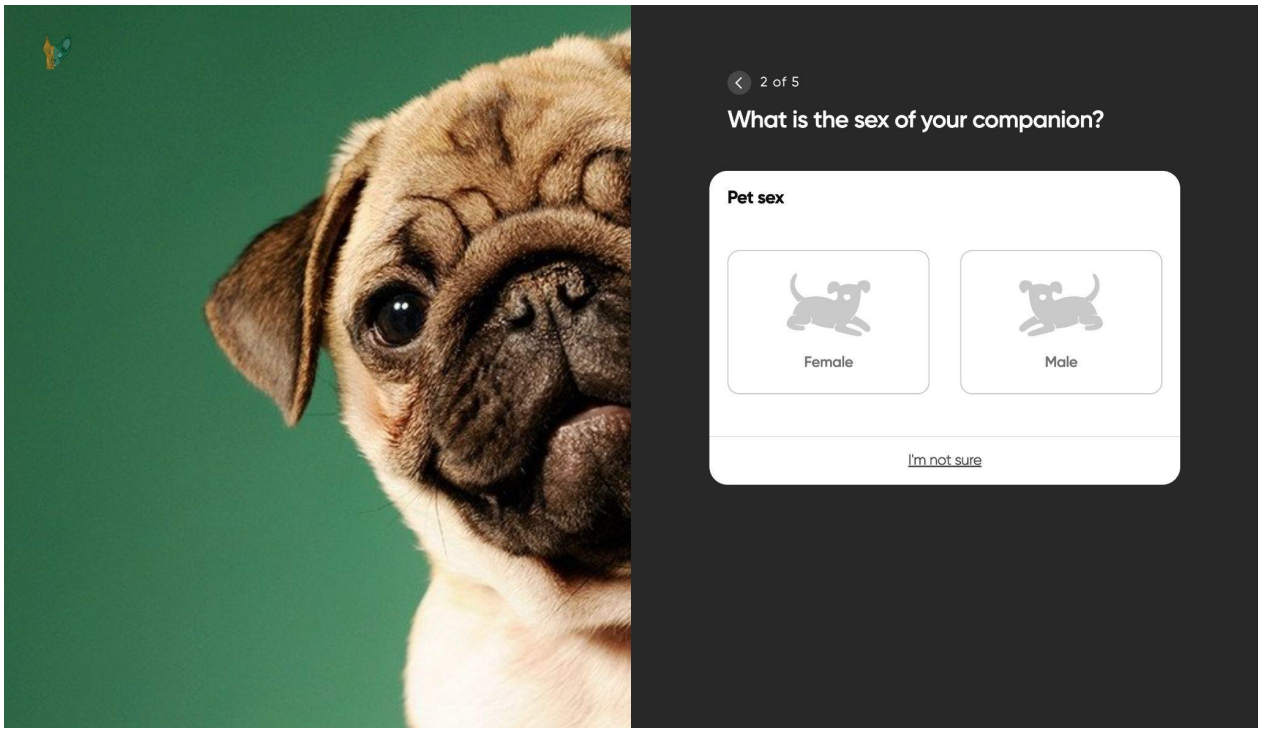


Рисунок 3.3 - другий крок додавання тварини

На третьому кроці треба заповнити ім’я, породу, дату народження - рисунок 3.4.

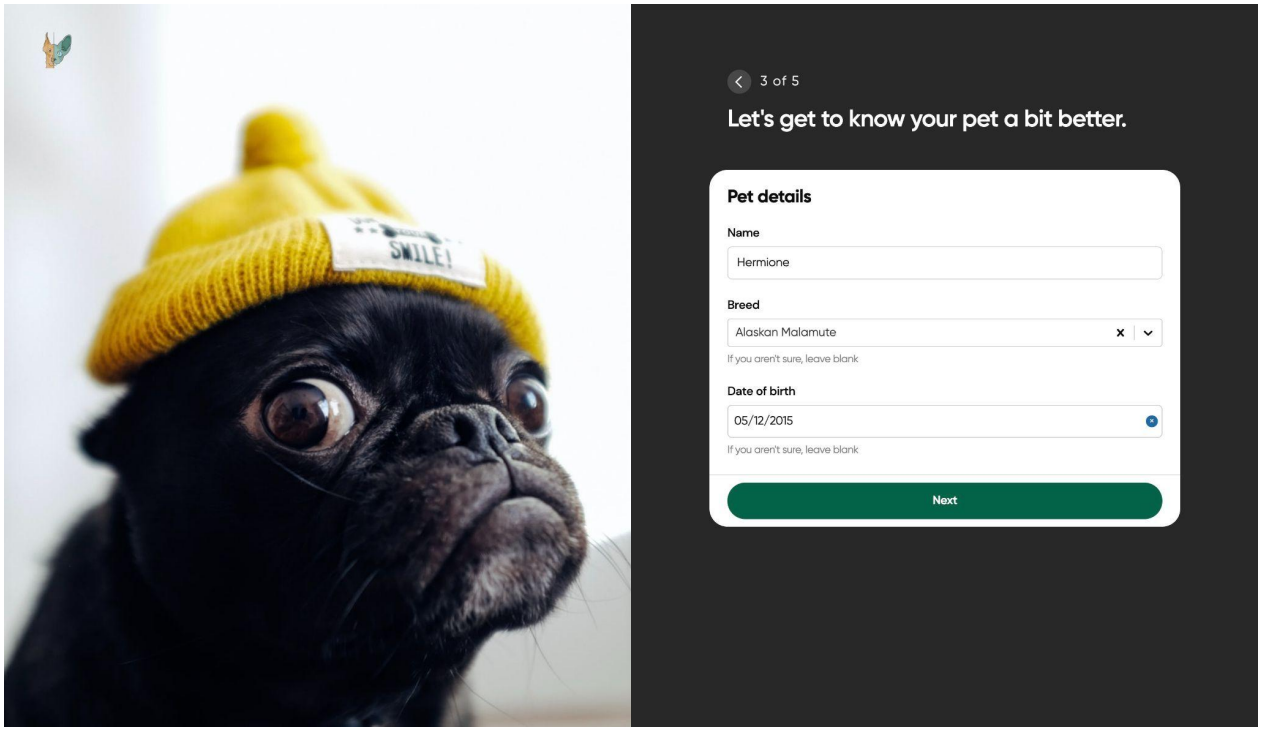


Рисунок 3.4 - третій крок додавання тварини

Четвертий крок для додавання фотографії тварини - рисунок 3.5.

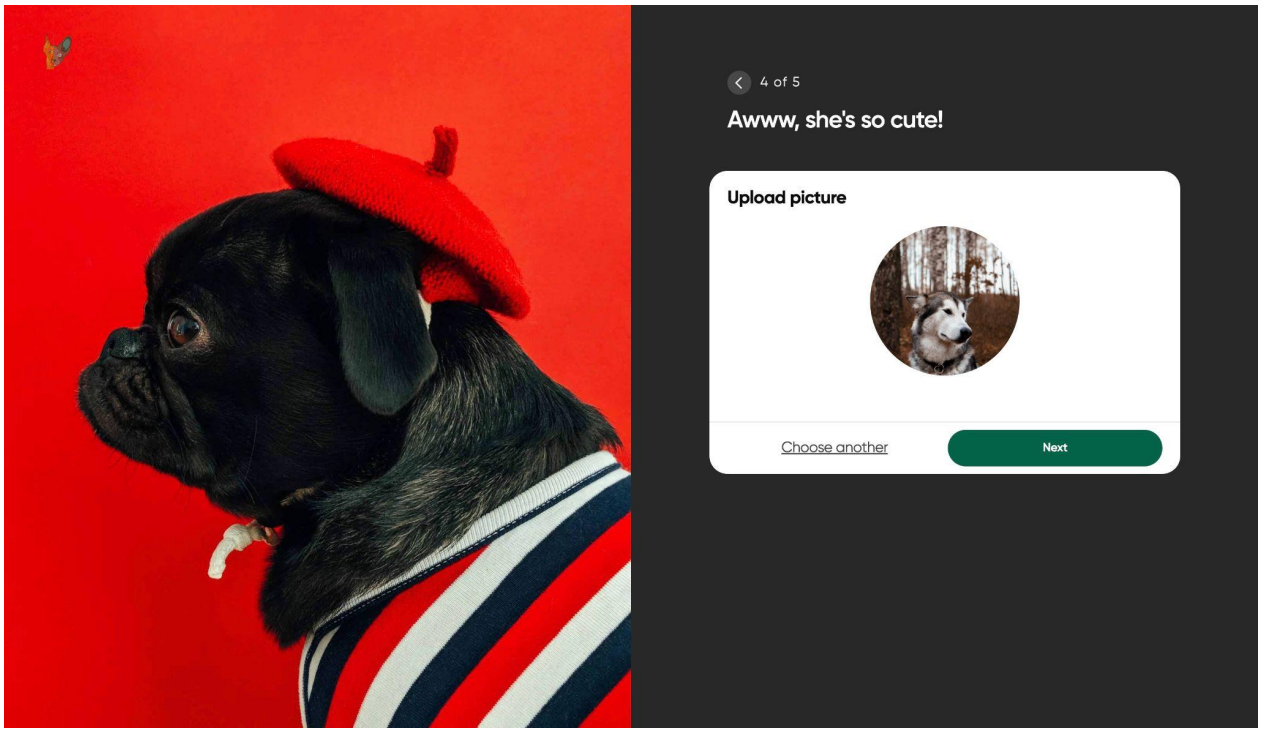


Рисунок 3.5 - четвертий крок додавання тварини

П’ятий крок - обрати улюблені ласощі тварини. Крок зображено на рисунку 3.6.

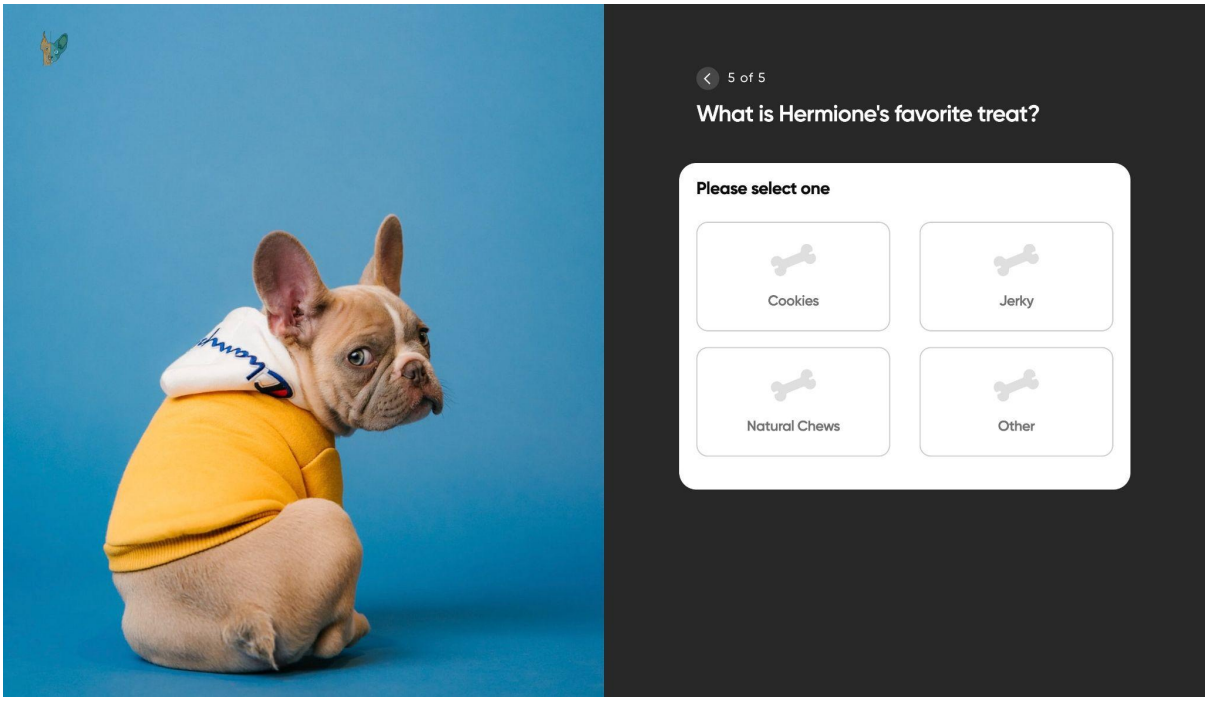


Рисунок 3.6 - п’ятий крок додавання тварини

По завершенню додавання відображається сторінка з привітанням - рисунок 3.7.

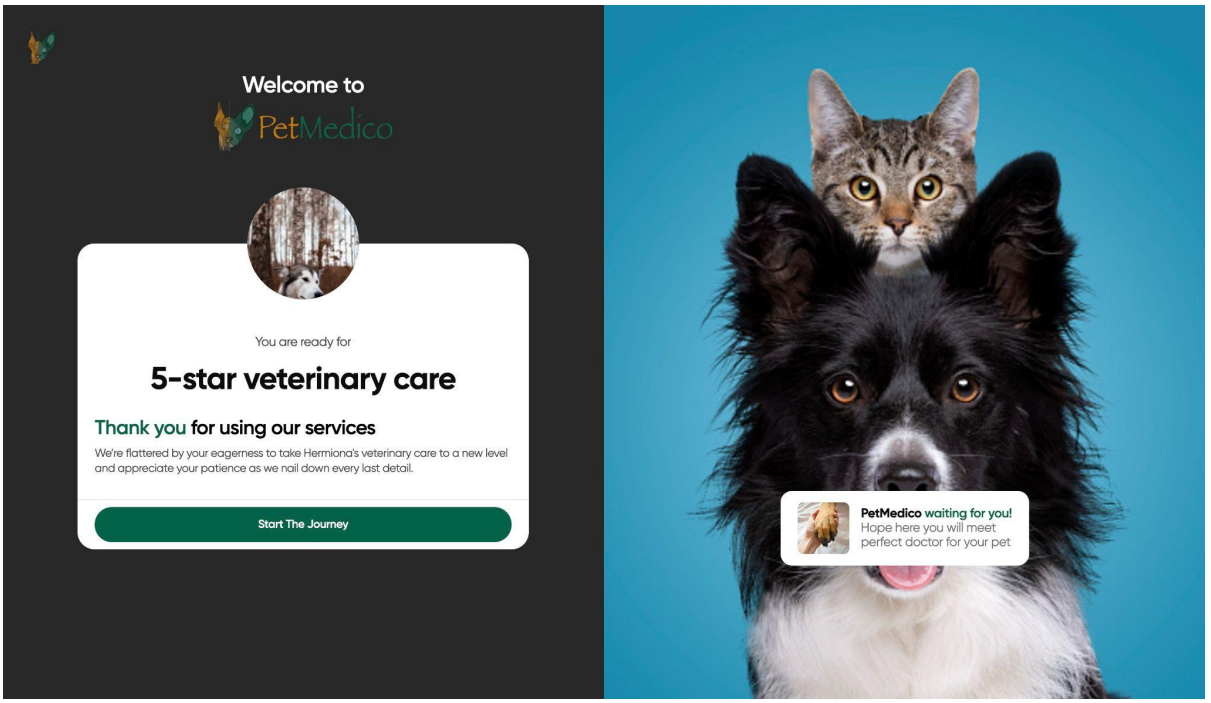


Рисунок 3.7 - вітання у системі

Користувач після реєстрації потрапляє до системи на сторінку налаштувань. Сторінка містить кнопки для переходу до двох секцій - доданих тварин та власного профілю. Сторінку зображено на рисунку 3.8.

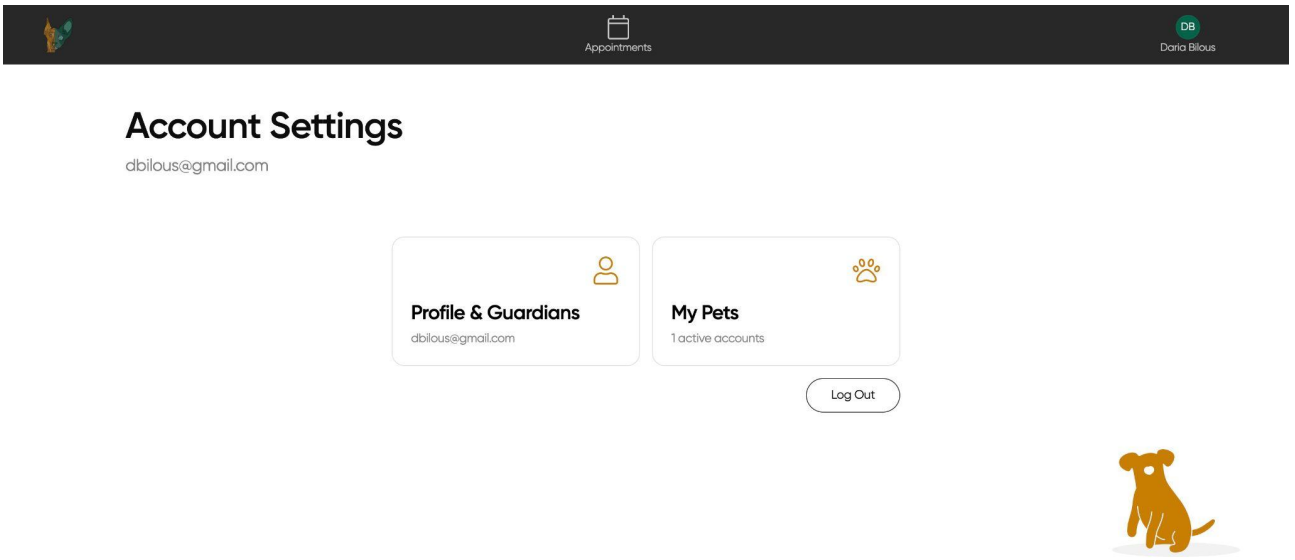


Рисунок 3.8 - сторінка налаштувань для користувача

Перейшовши до власного профілю користувач побачить сторінку із формою, що містить поля для редагування - рисунок 3.9

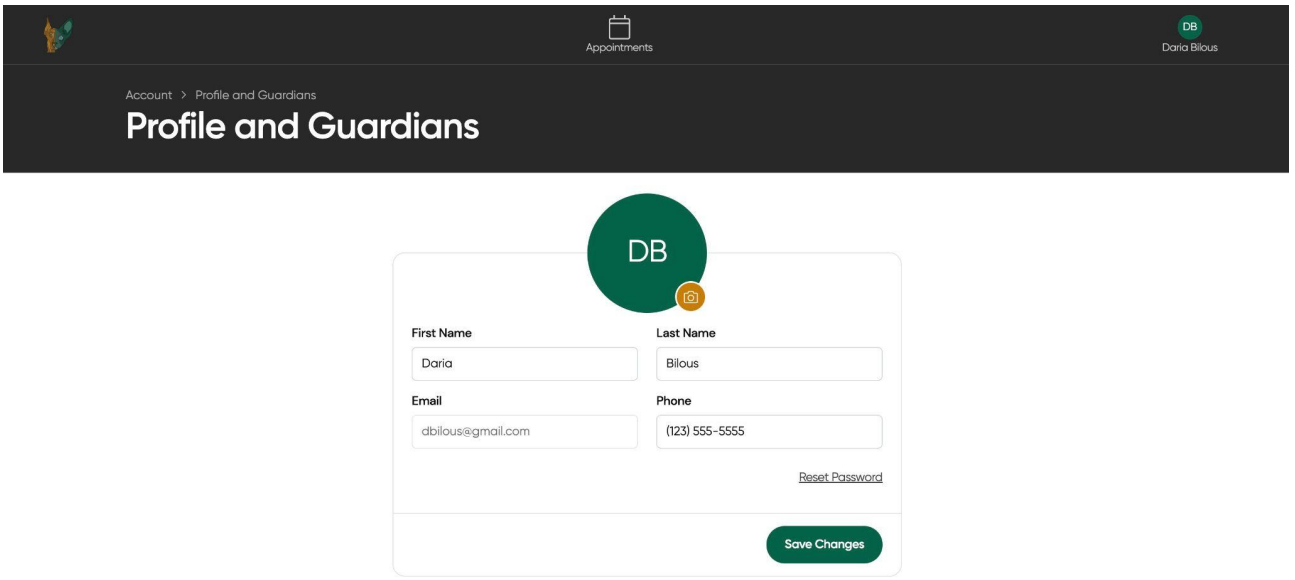


Рисунок 3.9 - профіль користувача

Перейшовши до секції тварин користувач побачить сторінку зі списком тварин у вигляді блоків та один блок для додавання нової - рисунок 3.10.

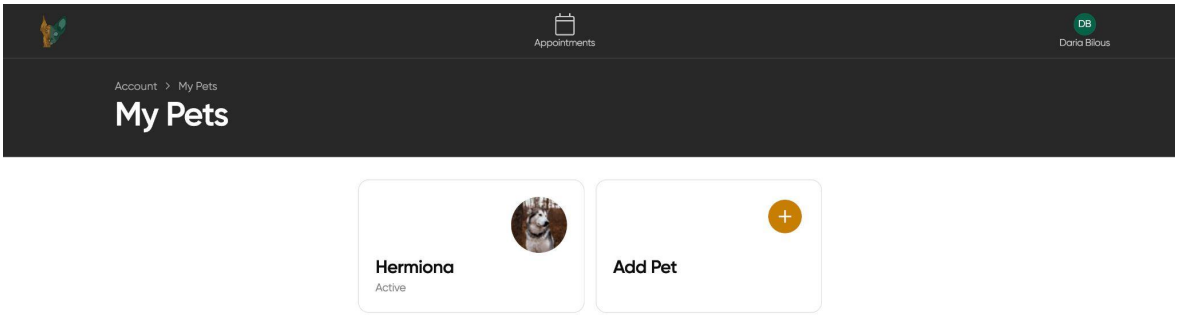


Рисунок 3.10 - сторінка тварин користувача

Натиснувши на лого системи у хедері користувач потрапить на дашборд. Поки немає жодного запису до лікаря, тож дашборд містить пропозицію створити її, а також пропозицію заповнити профіль тварини - рисунок 3.11.

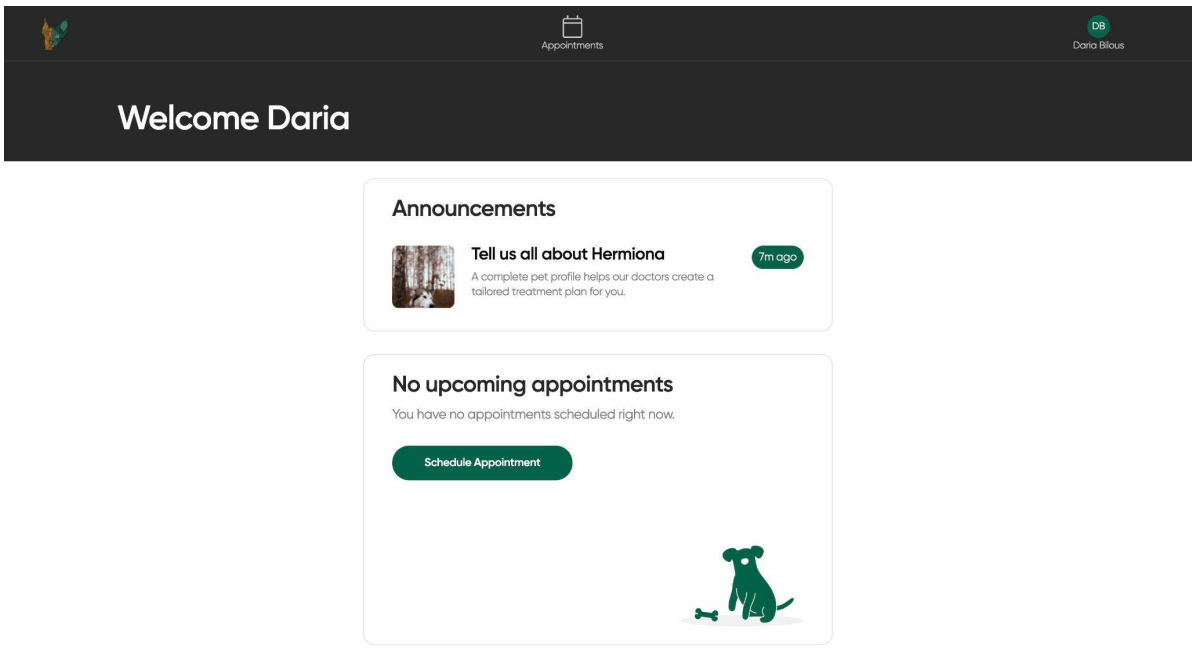


Рисунок 3.11 - дашборд користувача що не має жодного запису

Натиснувши на кнопку “Schedule appointment” користувач потрапить до сторінки створення запису до лікаря, створення запису має три кроки. Перший крок це вибір проблеми та додавання нотаток (за потребою) - рисунок 3.12.

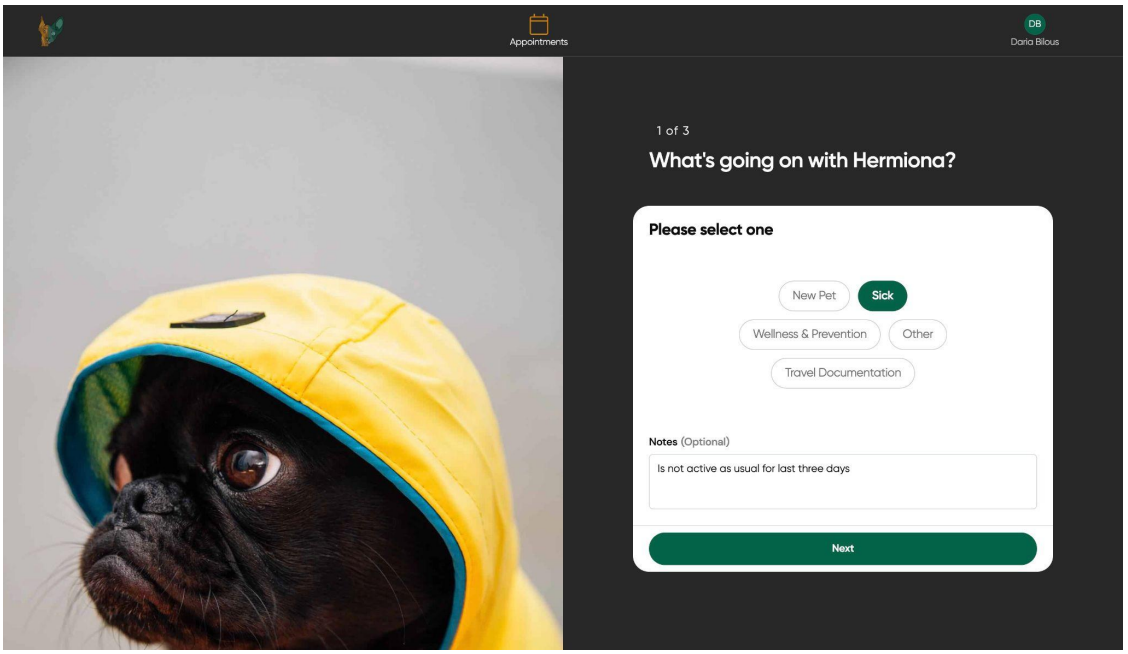


Рисунок 3.12 - перший крок створення запису до лікаря

На другому кроці потрібно обрати лікаря або натиснути “I’m flexible” якщо користувачу не принципово - рисунок 3.13.

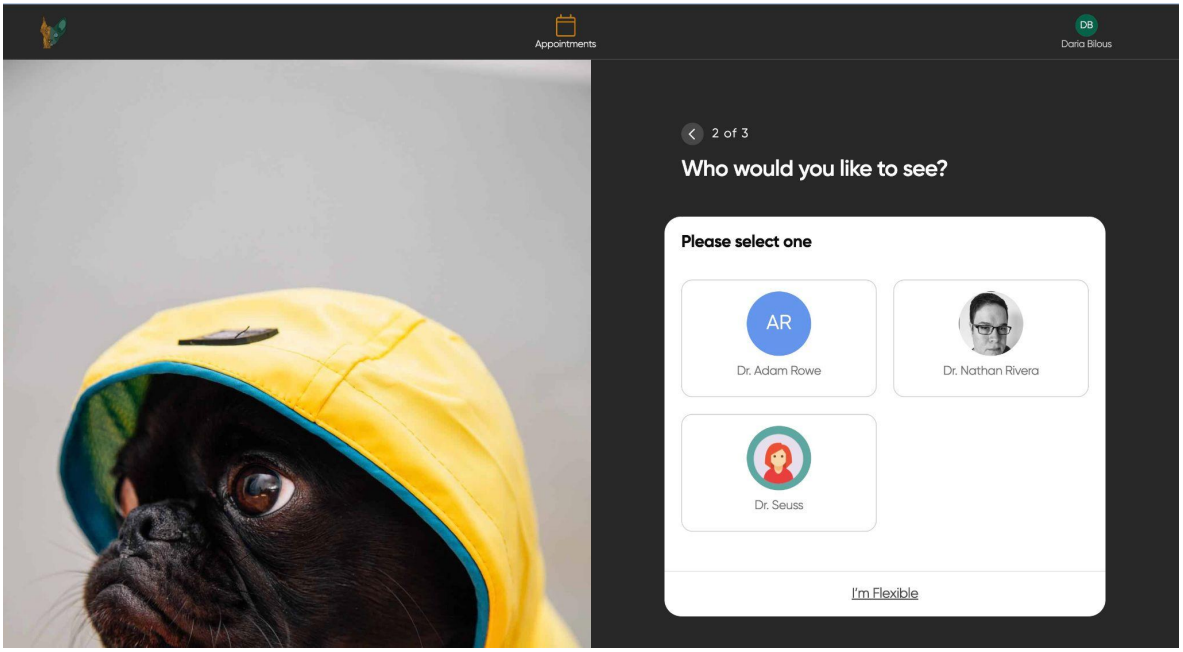


Рисунок 3.13 - другий крок створення запису до лікаря

Третій крок це вибір дати та часу - рисунок 3.14

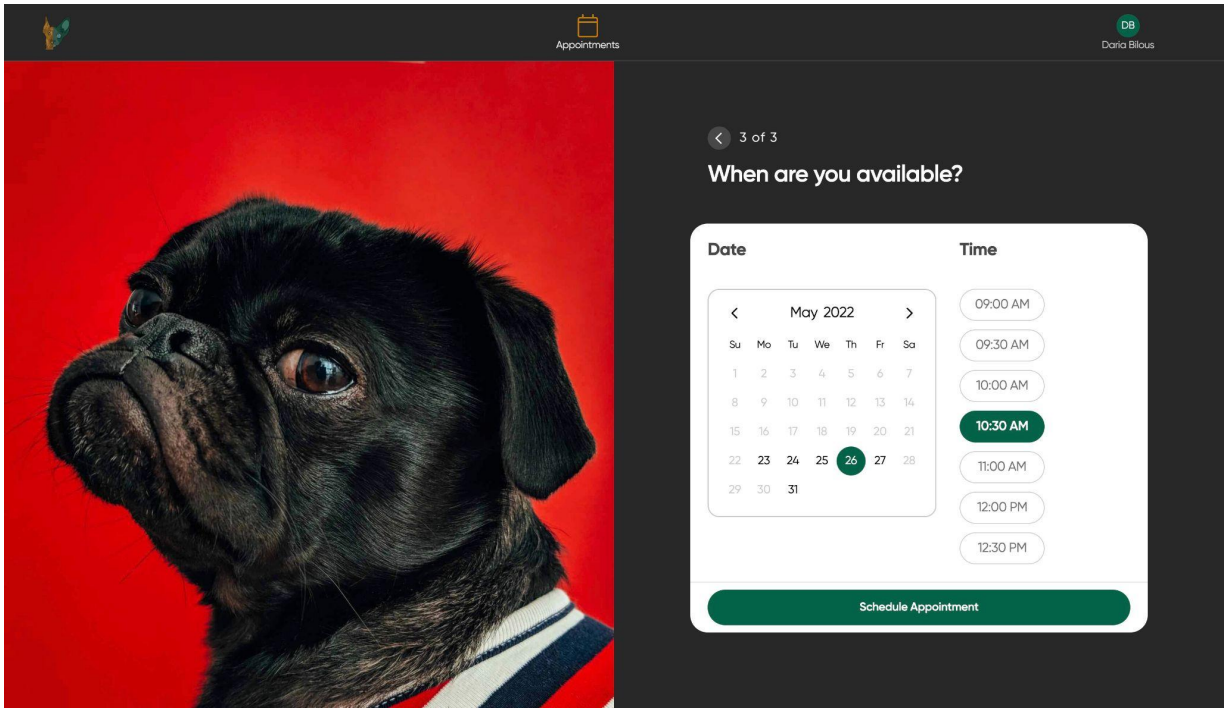


Рисунок 3.14 - третій крок створення запису до лікаря

Після завершення створення запису відображається сторінка з підтвердженням що запис створено - рисунок 3.15.

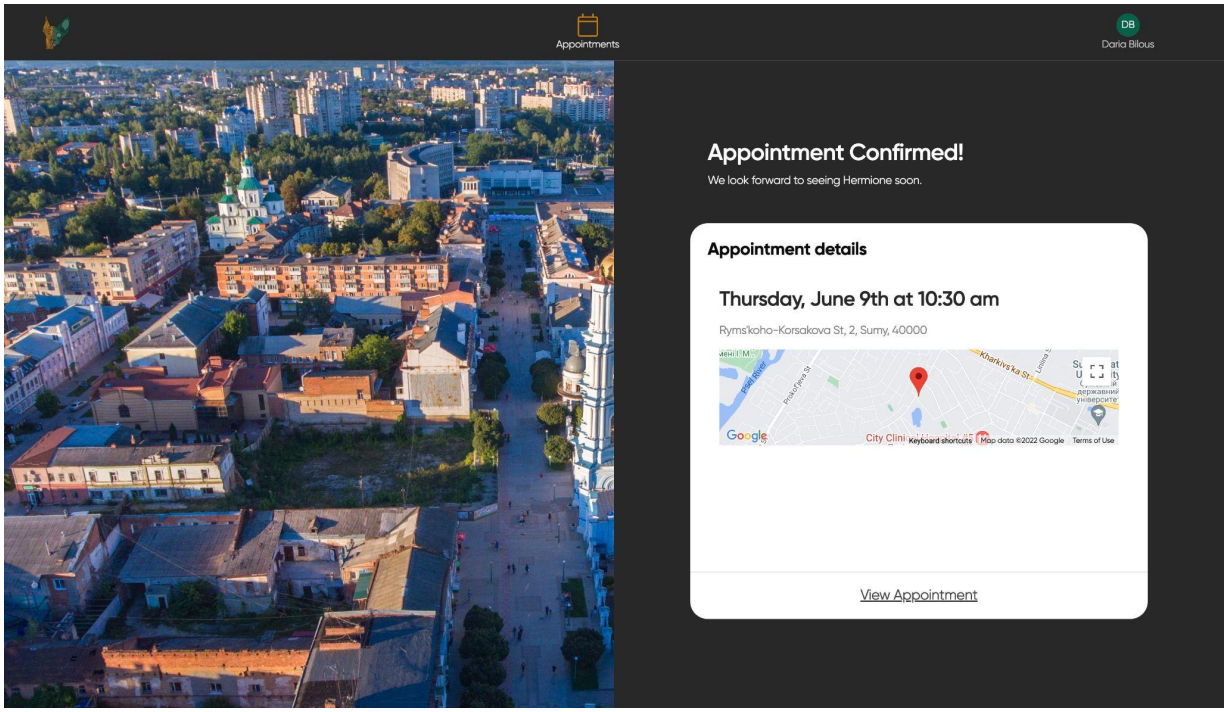


Рисунок 3.15 - запис до лікаря створено

Після створення запису на пошту приходить лист з деталями - рисунок 3.16

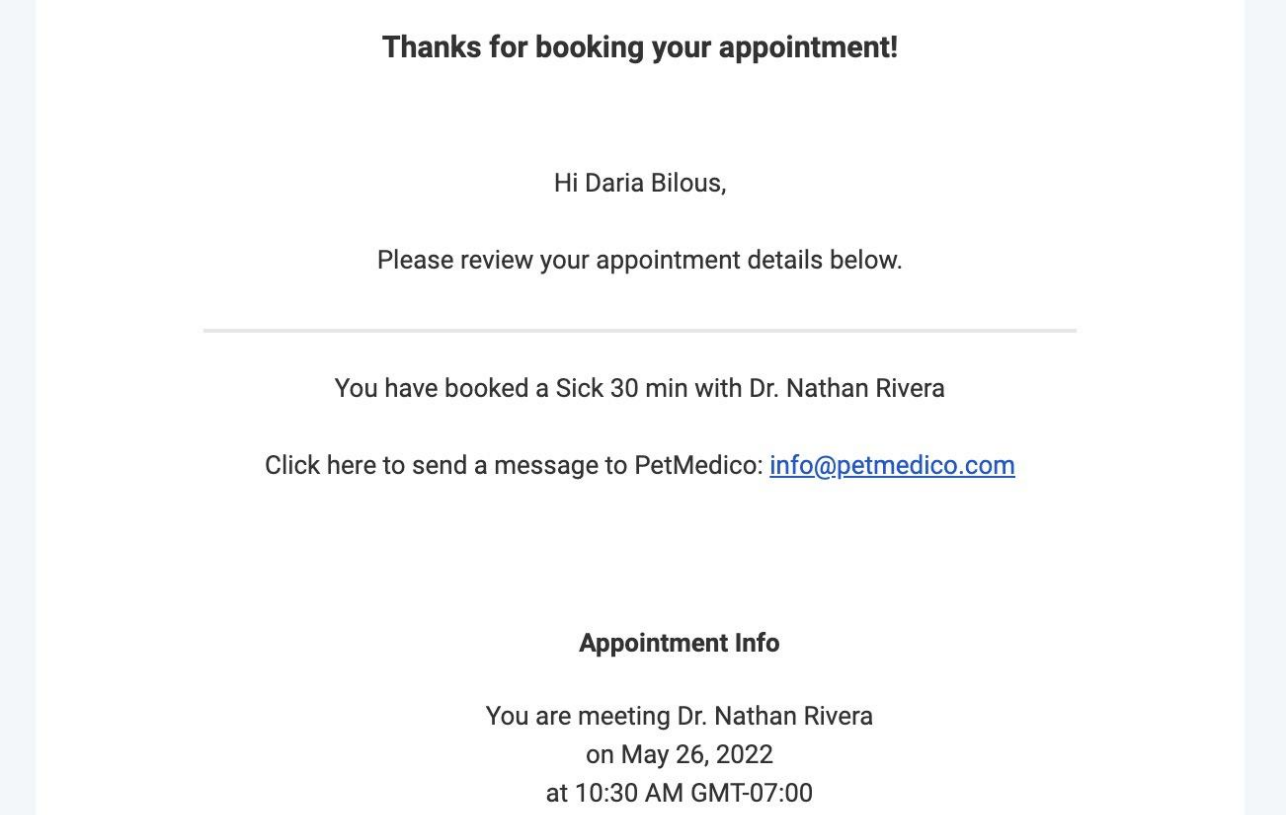


Рисунок 3.16 - лист з деталями створеного запису

Тепер якщо перейти до дашборду в секції записів буде створений запис - рисунок 3.17

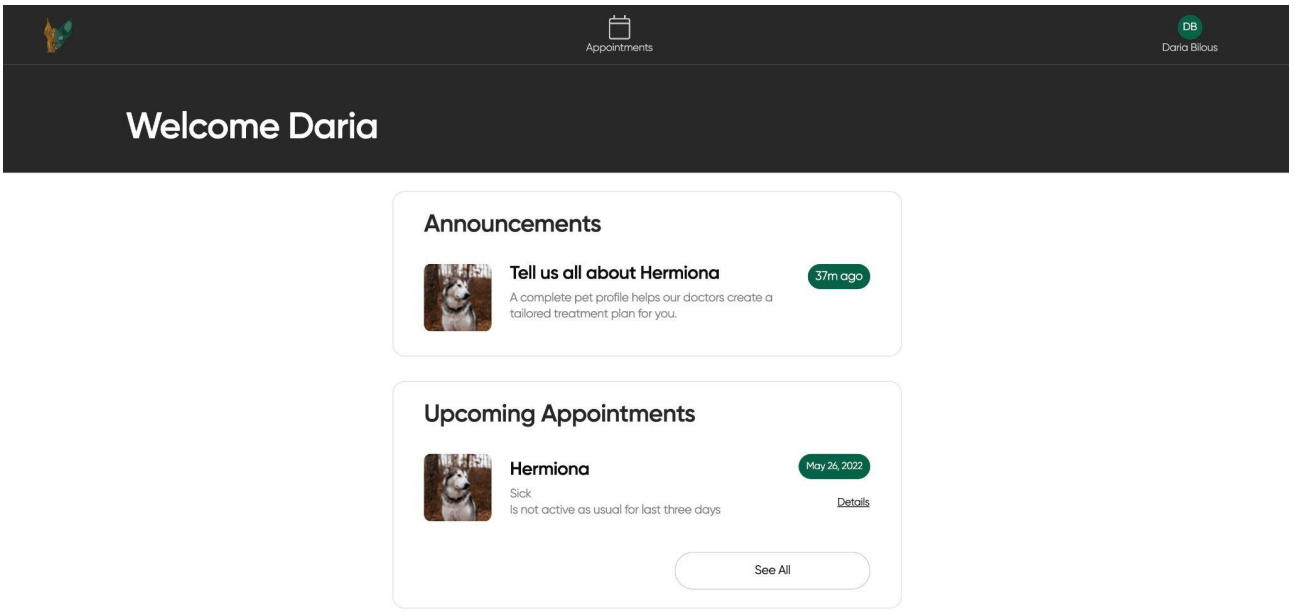


Рисунок 3.17 - дашборд із створеним записом

Натиснувши на пункт “Appointments” у хедері користувач потрапить на сторінку з записами. На ній будуть відображатися майбутні заплановані візити, можна буде перейти до створення нового візиту та переглянути попередні - рисунок 3.18.

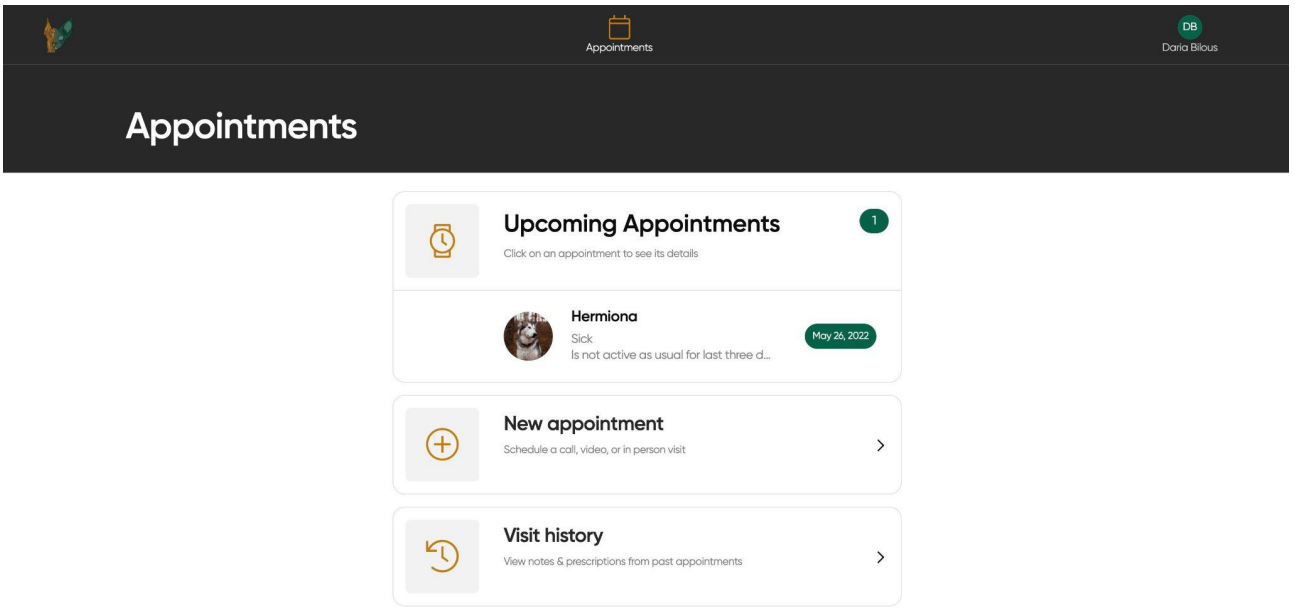


Рисунок 3.18 - сторінка записів

Зі списку записів можна перейти до деталей одного з них. Сторінка з деталями запису виглядає наступним чином: Рис. 3.19

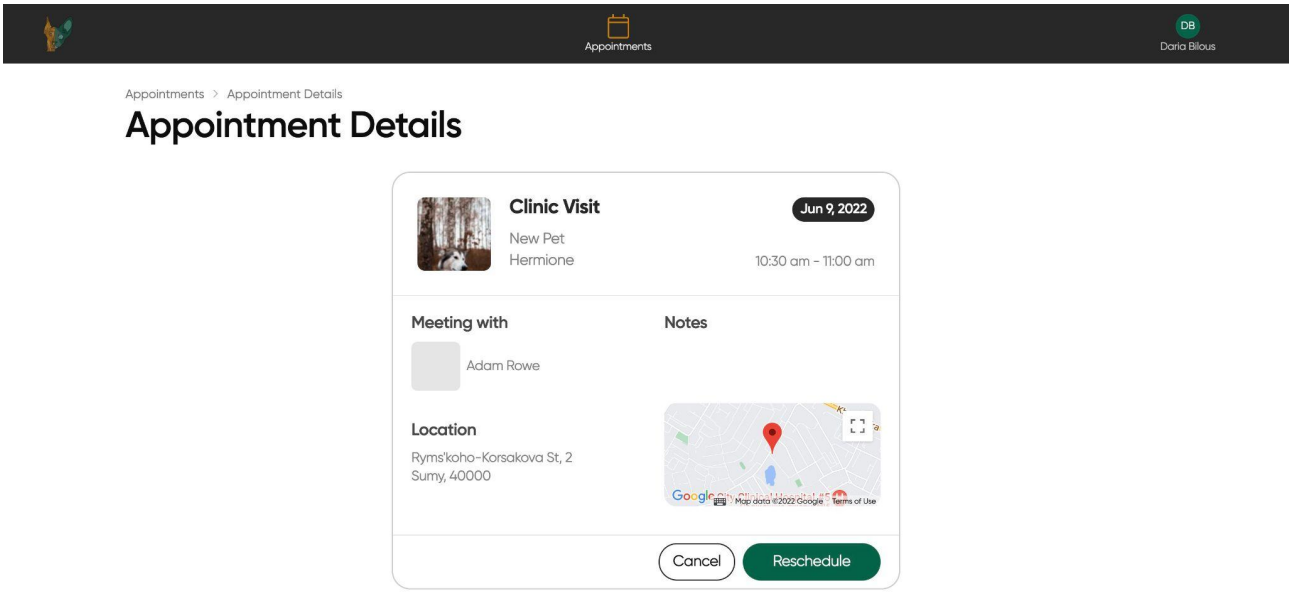


Рисунок 3.19 - деталі запису

Ще у системі є сторінка тварини. На ній власник може редагувати інформацію щоб вона була актуальною. Профіль тварини - рис. 3.20

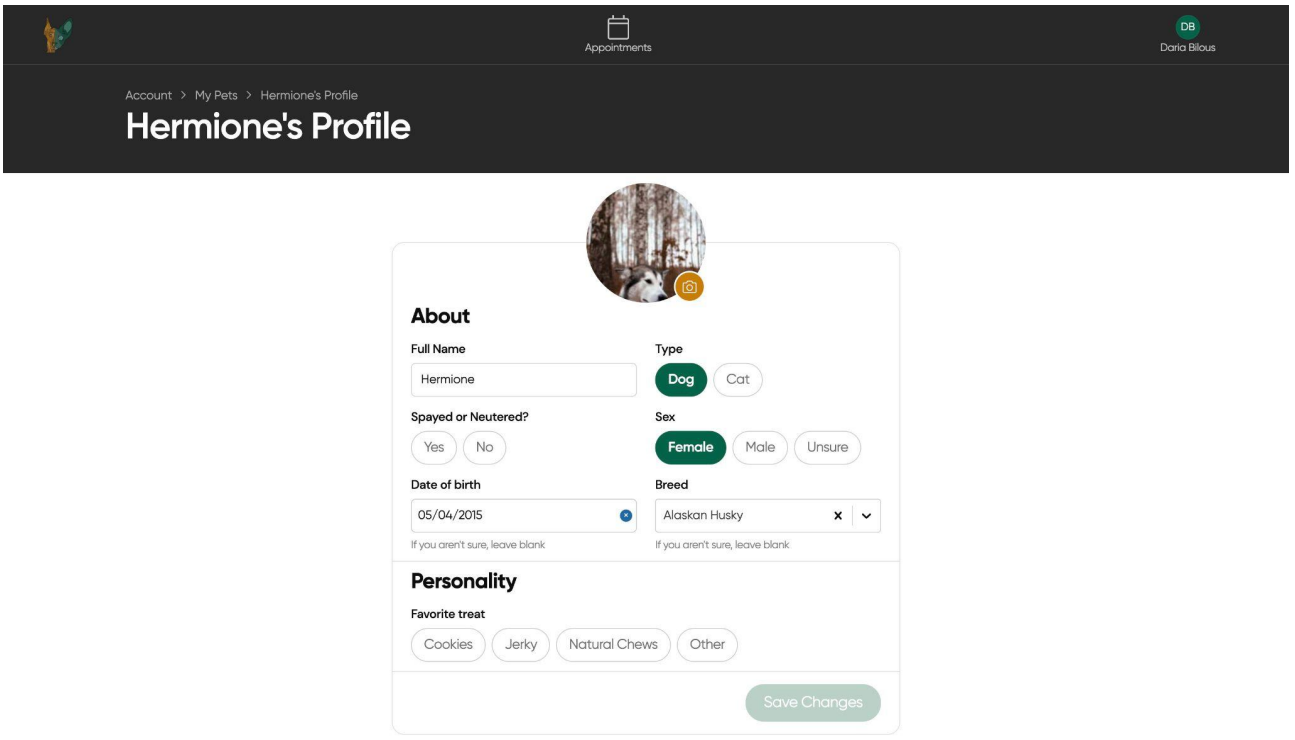


Рисунок 3.20 - сторінка тварини

Наразі було реалізовано інтерфейс для клієнта, інтерфейс для лікаря планується реалізувати у випускній роботі магістра. Крім описаного функціоналу є ідеї додати чат із лікарем та відео-візити, можливо зробити розділ з ліками, де можна буде прочитати інструкції та рекомендації до різних препаратів.

ВИСНОВКИ

Метою кваліфікаційної роботи є досягнення цілей та виконання завдань проекту, а саме розробка програмного забезпечення системи для організації роботи лікарів ветеринарної медицини та їх клієнтів.

Для виконання поставлених у роботі задач було проведено аналіз обраної предметної області та визначено актуальність проблеми. При проведенні даного аналізу було вивчено процеси взаємодії ветеринара і клієнта, виділені етапи які можливо автоматизувати. Було детально розглянуто процес онлайн бронювання візиту, різновиди бронювання, визначено переваги наявності ролі клієнта та в цілому використання системи автоматизації процесів у ветеринарній справі.

Проведено порівняльний аналіз аналогів для визначення їх переваг і недоліків. Головними відмінностями стали наявність ролі клієнта та придатність системи для використання приватними лікарями. Зробити систему саме для приватних ветеринарів, надати можливість людям самостійно керувати своєю інформацією та своїх тварин, створювати запис до лікаря онлайн і були головними ідеями проекту, тож він має такі переваги. Більшість існуючих застосунків розраховані на використання клініками, що не робить їх непридатними до використання приватними ветеринарами, але для такого сценарію є багато зайвого функціоналу та складних налаштувань.

Спираючись на отримані дані було сформовано функціональні вимоги до системи, створений її шаблон (мокап), обрані засоби реалізації. На етапі проектування веб-додатку були створені діаграми за методологіями IDEF та DFD для демонстрації структури і функцій системи, а також діаграма варіантів використання, де визначаються головні актори та сценарії їх взаємодії з компонентами інформаційної системи. Також було побудовано модель бази, яка допомогає легко візуалізувати її структуру. Не всі запропоновані функції були реалізовані в результаті виконання кваліфікаційної роботи бакалавра, але в базі даних закладена структура для більшості з них для подальшої реалізації у кваліфікаційній роботі магістра.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Методологія IDEF0: веб-сайт. URL: https://stud.com.ua/87184/ekonomika/metodologiya_idef0
2. UML - Use Case Diagrams : веб-сайт. URL: https://www.tutorialspoint.com/uml/uml_use_case_diagram.htm
3. Агольцов, В.А. Организация ветеринарного дела и экономика ветеринарных мероприятий / В.А. Агольцов, А.В. Красников. – Саратов, 2010. – 299 с.
4. HTML basics : веб-сайт. URL: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics
5. CSS Guides: веб-сайт. URL: https://css-tricks.com/guides/
6. Самовчитель CSS : веб-сайт. URL: http://htmlbook.ru/samcss
7. Ніксон Р. Створюємо динамічні веб-сайти за допомогою PHP, MySQL, JavaScript, CSS і HTML5, 2016. - 510 с.
8. Закон України про ветеринарну медицину : веб-сайт. URL: https://zakon.rada.gov.ua/laws/show/1206-20#Text
9. Шляхи розвитку медичної практики : веб-сайт. URL: https://www.myhealth1st.com.au/practices/resources/articles/10-ways-to-grow-your-medical-practice/
10. Писати від руки або друкувати все за і проти: веб-сайт. URL: https://jak.koshachek.com/articles/pisati-vid-ruki-abo-drukuvati-vse-za-i-proti-5.html
11. Дуглас Крокфорд JavaScript: The Good Parts: The Good Parts 1st Edition 2018 - 143 с.

ДОДАТОК А. ТЕХНІЧНЕ ЗАВДАННЯ

ТЕХНІЧНЕ ЗАВДАННЯ

на розробку програмного забезпечення системи для організації роботи лікарів ветеринарної медицини та їх клієнтів

Суми 2022

1 ПРИЗНАЧЕННЯ Й МЕТА СТВОРЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

1.1 Призначення інформаційної системи

Інформаційна система повинна забезпечити клієнтів - можливістю самостійно керувати даними тварин, обирати лікаря та бронювати візит онлайн, приватних лікарів ветеринарної медицини - можливістю автоматизувати процеси взаємодії з клієнтами.

1.2 Мета створення інформаційної системи

Основна мета інформаційної системи - вирішення проблеми доступу клієнтів (власників тварин) до медичної документації стосовно їх улюбленців, проблеми бронювання запису на прийом, проблеми передачі медичних призначень тварини між лікарями.

1.3 Цільова аудиторія

Цільова аудиторія – група людей, яка потенційно зацікавлена у послугах / сервісах які надає інформаційна система. У випадку розробляємого проекту це власники тварин, яким потрібна консультація ветеринара, та лікарі ветеринарної медицини, які працюють не в закладі та хочуть спростити ведення власної справи без створення власного застосунку.

2 ВИМОГИ ДО ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Вимоги до структури й функціонування інформаційної системи

Система повинна бути реалізована у вигляді Web-додатку, доступ до якого може отримати будь-який користувач маючи браузер та мережу Інтернет.

2.2 Вимоги до користувачів

Для роботи з даною інформаційною системою людині достатньо володіти загальними навичками роботи з персональним комп'ютером і вміти користуватися будь-яким стандартним веб-браузером.

2.3 Вимоги до стилістичного оформлення

Інтерфейс інформаційно системи має бути інтуїтивно зрозумілим, щоб користувач міг легко скласти уявлення про структуру наявної на сайті інформації, без проблем знайти як виконуються стандартні дії (наприклад бронювання запису). В інтерфейсі не має бути інших кольорів крім основних і акцентних (темно-сірий - *393939ff,* білий - *#ffffffff,* зелений - *#006349ff,* помаранчевий - *#c87e00ff*).

3 СТРУКТУРА ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Загальна інформація про структуру інформаційної системи

Структура системи являє собою набір сторінок, потрапити на будь-яку з яких можна за допомогою інтерфейсу додатку (наприклад натиснувши на пункт меню, або на лого системи у хедері).

Такими сторінками є:

Авторизація / реєстрація - має поля email і password та перемикач Sign In / Sign Up в залежності від бажаної дії

Додавання тварини – має фото зліва, та блок для питань та відповідей справа

Успішна реєстрація – має фото справа, та блок із загальною інформацією зліва

Сторінка налаштувань – містить два блоки, один з яких веде до профілю користувача, а інший до сторінки з тваринами

Профіль користувача – містить форму з полями для особистих даних користувача та його фото

Сторінка з тваринами – містить блоки, кожен з яких відповідає тварині, яку додав власник. Останнім блоком є блок для додавання нової тварини

Профіль тварини – містить форму з полями для особистих даних тварини, та її фото

Дашборд – може містити нагадування (наприклад доповнити профіль тварини), містить список запланованих записів

Онлайн бронювання – містить фото зліва, справа блок для інформації, необхідної для створення бронювання

Записи - стороінка містить заплановані, минулі записи для блок для створення нового запису

Деталі запису - містить інформацію про запис (таку як тварина, дата, причина візиту, адреса, нотатки)

ДОДАТОК Б. ЛІСТИНГ КОДУ

```
/ front entry point /
import { StrictMode } from 'react';
import * as ReactDOM from 'react-dom';
import '@stripe/stripe-js';
import { configureMoment } from '@petmedico/common/utils/moments';
import { requestService } from
'@petmedico/common/network/services/RequestService';
import { resetQueriesToUndefined } from '@petmedico/common/network/utils/query';
import { LocalValueStorage } from './app/utils/LocalValueStorage';

import { App } from './app/@core/App';

import '@petmedico/ui-web/styles/global.scss';

configureMoment();

requestService.setStorage(new LocalValueStorage());
requestService.setCallbacks({
 onUnauthorized: () => {
   resetQueriesToUndefined();
 },
});

ReactDOM.render(
 <StrictMode>
   <App />
 </StrictMode>,
 document.getElementById('root')
);

/ react app main /
import React from 'react';
import { QueryClientProvider } from 'react-query';
import { ProfileQueryProvider } from
'@petmedico/common/api/profile/contexts/ProfileQueryContext';
import { ToastProvider } from '@petmedico/ui-web/components/Toast';
import { queryClient } from '@petmedico/common/network/queryClient';

import { AppRoutes } from '../routes/AppRoutes';
```

```
export const App: React.VFC = () => (
 <QueryClientProvider client={queryClient}>
   <ProfileQueryProvider>
     <ToastProvider>
       <AppRoutes />
     </ToastProvider>
   </ProfileQueryProvider>
 </QueryClientProvider>
);

/ router /
/ 1 /
import React from 'react';
import { BrowserRouter } from 'react-router-dom';
import { useProfileQuery } from
'@petmedico/common/api/profile/queries/useProfileQuery';

import { PublicRoutes } from './PublicRoutes';
import { RestrictedRoutes } from './RestrictedRoutes';

export const AppRoutes: React.VFC = () => {
 const { profile, isLoading } = useProfileQuery();
 if (isLoading) {
   return null;
 }
 const isAuthenticated = profile !== undefined;
 return <BrowserRouter>{isAuthenticated ? <RestrictedRoutes /> : <PublicRoutes
/>}</BrowserRouter>;
};

/ 2 /
import React from 'react';
import { Redirect, Route, Switch } from 'react-router-dom';
import { AccountSettingsPageContainer } from
'../../account/@core/AccountSettingsPageContainer';
import { ProfilePageContainer } from '../../account/@core/ProfilePageContainer';
import { PetsContainer } from '../../account/@core/PetsContainer';
import { PetProfileContainer } from '../../account/@core/PetProfileContainer';
import { accountRoutes } from '../../account/const/accountRoutes';

export const AccountRoutes: React.VFC = () => {
 return (
```

```
   <Switch>
     <Route
       exact
       path={accountRoutes.Account}
       component={AccountSettingsPageContainer}
     />
     <Route
       exact
       path={accountRoutes.Profile}
       component={ProfilePageContainer}
     />
     <Route
       exact
       path={accountRoutes.Pets}
       component={PetsContainer}
     />
     <Route
       exact
       path={accountRoutes.PetProfile}
       component={PetProfileContainer}
     />
     <Redirect to={accountRoutes.Account} />
   </Switch>
 );
};

/ 3 /
import React, { useEffect } from 'react';
import { Redirect, Route, Switch } from 'react-router-dom';
import { useClearNewAppointmentMutation } from
'@petmedico/common/api/appointments/mutations/useClearNewAppointmentMutation';
import { AppointmentPageContainer } from
'../../appointments/@core/AppointmentPageContainer';
import { CalendarPageContainer } from
'../../appointments/@core/CalendarPageContainer';
import { ChooseCareTypePageContainer } from
'../../appointments/@core/ChooseCareTypePageContainer';
import { ReasonPageContainer } from
'../../appointments/@core/ReasonPageContainer';
import { SummaryPageContainer } from
'../../appointments/@core/SummaryPageContainer';
import { VetPageContainer } from '../../appointments/@core/VetPageContainer';
```

```
import { newAppointmentRoutes } from '../../appointments/const/appointmentRoutes';

export const NewAppointmentRoutes: React.VFC = () => {
 const { clearAppointment } = useClearNewAppointmentMutation();

 useEffect(() => {
   return () => {
     clearAppointment();
   };
 }, [clearAppointment]);

 return (
   <Switch>
     <Route
       exact
       path={newAppointmentRoutes.NewAppointment}
       component={AppointmentPageContainer}
     />
     <Route
       exact
       path={newAppointmentRoutes.NewAppointmentReason}
       component={ReasonPageContainer}
     />
     <Route
       exact
       path={newAppointmentRoutes.NewAppointmentCareType}
       component={ChooseCareTypePageContainer}
     />
     <Route
       exact
       path={newAppointmentRoutes.NewAppointmentVet}
       component={VetPageContainer}
     />
     <Route
       exact
       path={newAppointmentRoutes.NewAppointmentCalendar}
       component={CalendarPageContainer}
     />
     <Route
       exact
       path={newAppointmentRoutes.NewAppointmentSummary}
       component={SummaryPageContainer}
```

```
      />
      <Redirect to={newAppointmentRoutes.NewAppointment} />
    </Switch>
 );
};

/ 4 /
import React from 'react';
import { Redirect, Route, Switch } from 'react-router-dom';
import { AppointmentsPageContainer } from
'../../dashboard/@core/AppointmentsPageContainer';
import { AppointmentDetailsPageContainer } from
'../../dashboard/@core/AppointmentDetailsPageContainer';
import { HomePageContainer } from '../../dashboard/@core/HomePageContainer';
import { AppView } from '../components/AppView';
import { AppRoute } from '../const/AppRoute';
import { AccountRoutes } from './AccountRoutes';
import { NewAppointmentRoutes } from './NewAppointmentRoutes';

export const SecuredRoutes: React.VFC = () => (
 <AppView>
   <Switch>
     <Route
       exact
       path={AppRoute.Dashboard}
       component={HomePageContainer}
     />
     <Route
       path={AppRoute.Account}
       component={AccountRoutes}
     />
     <Route
       exact
       path={AppRoute.Appointments}
       component={AppointmentsPageContainer}
     />
     <Route
       path={AppRoute.NewAppointment}
       component={NewAppointmentRoutes}
     />
     <Route
       path={AppRoute.AppointmentDetails}
```

```
        component={AppointmentDetailsPageContainer}
      />
      <Redirect to={AppRoute.Dashboard} />
    </Switch>
  </AppView>
);

/ 5 /
import React, { useEffect } from 'react';
import { Route, Switch, useLocation } from 'react-router-dom';
import { useProfileQuery } from
'@petmedico/common/api/profile/queries/useProfileQuery';

import { AppRoute } from '../const/AppRoute';
import { OnboardingRoutes } from './OnboardingRoutes';
import { SecuredRoutes } from './SecuredRoutes';

export const RestrictedRoutes: React.VFC = () => {
  const { profile } = useProfileQuery();
  const { pathname } = useLocation();

  useEffect(() => {
    window.scrollTo(0, 0);
  }, [pathname]);

  const isFinishedOnboardingSteps = profile?.current_onboarding_step ===
'finished';

  return (
    <Switch>
      {isFinishedOnboardingSteps ? (
        <Route
          path={AppRoute.Home}
          component={SecuredRoutes}
        />
      ) : (
        <Route
          path={AppRoute.Home}
          component={OnboardingRoutes}
        />
      )}
    </Switch>
```

```
 );
};

/ 6 /
import React, { useEffect } from 'react';
import { Redirect, Route, Switch } from 'react-router-dom';
import { useHistory } from 'react-router';
import { useProfileQuery } from
'@petmedico/common/api/profile/queries/useProfileQuery';
import { onboardingRoutes, ONBOARDING_STEP } from
'../../onboarding/const/onboardingRoutes';
import { LocationPageContainer } from
'../../onboarding/@core/LocationPageContainer';
import { PetDetailsPageContainer } from
'../../onboarding/@core/PetDetailsPageContainer';
import { PetFavoriteTreatPageContainer } from
'../../onboarding/@core/PetFavoriteTreatPageContainer';
import { PetPictureUploadPageContainer } from
'../../onboarding/@core/PetPictureUploadPageContainer';
import { PetSexPageContainer } from '../../onboarding/@core/PetSexPageContainer';
import { PetSpeciesPageContainer } from
'../../onboarding/@core/PetSpeciesPageContainer';
import { PrelaunchNotificationPageContainer } from
'../../onboarding/@core/PrelaunchNotificationPageContainer';

export const OnboardingRoutes: React.VFC = () => {
 const { profile } = useProfileQuery();
 const { push } = useHistory();

 useEffect(() => {
   if (profile?.current_onboarding_step) {
     let pathname = ONBOARDING_STEP[profile.current_onboarding_step];
     if (pathname === undefined) {
       console.error(`Path does not exist: ${pathname}`);
       pathname = onboardingRoutes.Location;
     }
     push({ pathname });
   }
 }, [profile, push]);

 return (
   <Switch>
```

```
      <Route
        exact
        path={onboardingRoutes.Location}
        component={LocationPageContainer}
      />
      <Route
        exact
        path={onboardingRoutes.PetSpecies}
        component={PetSpeciesPageContainer}
      />
      <Route
        exact
        path={onboardingRoutes.PetSex}
        component={PetSexPageContainer}
      />
      <Route
        exact
        path={onboardingRoutes.PetDetails}
        component={PetDetailsPageContainer}
      />
      <Route
        exact
        path={onboardingRoutes.PetPictureUpload}
        component={PetPictureUploadPageContainer}
      />
      <Route
        exact
        path={onboardingRoutes.PetFavoriteTreat}
        component={PetFavoriteTreatPageContainer}
      />
      <Route
        exact
        path={onboardingRoutes.PrelaunchNotification}
        component={PrelaunchNotificationPageContainer}
      />
      <Redirect to={onboardingRoutes.Location} />
    </Switch>
  );
};

/ 7 /
import React from 'react';
```

```
import { Redirect, Route, Switch } from 'react-router-dom';
import { Login } from '../../login/@core/Login';
import { ResetPassword } from '../../login/@core/ResetPassword';
import { AppRoute } from '../const/AppRoute';

export const PublicRoutes: React.VFC = () => (
 <Switch>
   <Route
     path={AppRoute.Login}
     component={Login}
   />
   <Route
     path={AppRoute.ResetPassword}
     component={ResetPassword}
   />
   <Redirect to={AppRoute.Login} />
 </Switch>
);
```

```
/ api entry point /
require_relative 'boot'

require 'rails/all'
require 'good_job/engine'

# Require the gems listed in Gemfile, including any gems
# you've limited to :test, :development, or :production.
Bundler.require(*Rails.groups)

module Petmedico
  class Application < Rails::Application
    # Initialize configuration defaults for originally generated Rails version.
    config.load_defaults 7.0

    config.autoload_paths += Dir.glob("#{config.root}/app/interactions/*")

    # Configuration for the application, engines, and railties goes here.
    #
    # These settings can be overridden in specific environments using the files
    # in config/environments, which are processed later.
    #
    config.active_record.default_timezone = :utc
```

```
    config.time_zone = 'UTC'
    # config.eager_load_paths << Rails.root.join("extras")

    # Only loads a smaller set of middleware suitable for API only apps.
    # Middleware like session, flash, cookies can be added back manually.
    # Skip views, helpers and assets when generating a new resource.
    config.api_only = true

    # Needed for OmniAuth, see https://github.com/omniauth/omniauth#rails-api
    config.session_store :cookie_store, key: '_petmedico_session', skip: true
    config.middleware.use ActionDispatch::Cookies # Required for all session
management
    config.middleware.use ActionDispatch::Session::CookieStore,
config.session_options
  end
end

/ routes /
Rails.application.routes.draw do
  resources :animals, except: [:destroy]
  resources :appointments, only: %i[index show create] do
    collection do
      get :available_days
      get :available_times
      get :resources
      get :services
    end
    member do
      put :cancel
      put :reschedule
    end
  end
  resources :breeds, only: [:index]
  resources :consults, only: [:index]
  resources :dashboard, controller: :dashboard, only: [:index]
  resources :medical_records, only: [:show]
  resources :prescriptions
  resources :presenting_problems
  resources :vaccinations

  devise_for :customers,
             defaults: { format: :json },
```

```
              path: '',
              path_names: {
                sign_in: 'login',
                sign_out: 'logout',
                registration: 'signup'
              },
              controllers: {
                sessions: 'customers/sessions',
                registrations: 'customers/registrations',
                passwords: 'customers/passwords'
              }
  get 'profile', to: 'customers#show'
  put 'profile', to: 'customers#update'
end

/ appointment /
class Appointment < ApplicationRecord
  belongs_to :animal
  belongs_to :employee
  belongs_to :visit_reason
  has_one :consults

  scope :not_canceled, -> { where('canceled_at is null') }
  scope :upcoming, -> { where('start_time between ? and ?', Time.now, Time.now +
1.year) }
  scope :by_start_date, -> { order(start_time: :asc) }

  acts_as_taggable_on :visit_reasons

  def self.visit_reason_options(service_name)
    case service_name
    when 'Wellness'
      ['New Pet', 'Puppy/kitten', 'Bi-annual wellness exam']
    when 'Sick Pet or Illness'
      ['Itching or skin issue',
       'Ear issue',
       'Eye problem',
       'Vomiting',
       'Diarrhea or stool change',
       'Coughing or Sneezing',
       'Lethargy',
       'Lump/Bump',
```

```
        'Urinary issue',
        'Fleas, ticks or worms',
        'Injuries or Wound',
        'Behavioral change',
        'Pain or limping']
    when 'Preventive & Diagnostics'
      ['Vaccines',
        'Bloodwork/ Urine',
        'Blood pressure',
        'Anal gland expression',
        'Cytopoint injection',
        'Adequan injection']
    else
      []
    end
  end
end

/ animal /
class Animal < ApplicationRecord
  belongs_to :breed, optional: true
  belongs_to :customer

  has_many :appointments
  has_many :health_statuses
  has_many :prescriptions

  validate :species_compatible_breed

  def species_compatible_breed
    if breed.present? && breed.species_id != species_id
      errors.add(:breed_id, 'must have a matching species')
    end
  end

  def complete_profile?
    desexed.present? && weight.present?
  end

  def dog?
    species.name == Species::DOG
  end
```

```
  def cat?
    !dog?
  end
end

/ appointment controller /
class AppointmentsController < ApplicationController
  before_action :authenticate_customer!

  def index
    render_success(record: current_customer.appointments.by_start_date.map { |a|
appointment_json(a) })
  end

  def show
    if appointment = current_customer.appointments.find_by_id(params[:id])
      render_success(record: appointment_json(appointment))
    else
      render_error(message: 'Appointment not found')
    end
  end

  def create
    animal = current_customer.animals.find(params[:animal_id])
    appointment = Scheduling::Api.new.create_appointment(animal:, notes:
params[:notes],
service_id: params[:service_id],
resource_id: params[:resource_id],
start_date_time: params[:start_date_time],
end_date_time: params[:end_date_time],
complete_booking: params[:complete_booking],
visit_reason_list: params[:visit_reason_list])
render_success(record: appointment)
  end

  def cancel
    appointment = current_customer.appointments.find(params[:id])
    result = Scheduling::Api.new.cancel_appointment(scheduling_provider_id:
appointment.scheduling_provider_id)
    render_success(record: result)
  end
```

```
  def reschedule
    appointment = current_customer.appointments.find(params[:id])
    result = Scheduling::Api.new.reschedule_appointment(scheduling_provider_id:
appointment.scheduling_provider_id,
service_id: params[:service_id],
resource_id: params[:resource_id],
start_date_time: params[:start_date_time],
end_date_time: params[:end_date_time])
    render_success(record: result)
  end

  def services
    render_success(record: services_with_visit_reasons)
  end

  def resources
    render_success(record: Scheduling::Api.new.resources(service_id:
params[:service_id]))
  end

  def available_days
    result = Scheduling::Api.new.available_days(service_id: params[:service_id],
                                                resource_id: params[:resource_id],
                                                start_date: params[:start_date],
                                                end_date: params[:end_date])
    render_success(record: result)
  end

  def available_times
    result = Scheduling::Api.new.available_times(service_id: params[:service_id],
                                                 resource_id:
params[:resource_id],
                                                 start_date: params[:start_date])
    render_success(record: result)
  end

  private

  def services_with_visit_reasons
    Scheduling::Api.new.services.map do |service|
```

```
      service.merge(visitReasons:
Appointment.visit_reason_options(service[:name]))
    end
  end

  def appointment_json(appointment)
    AppointmentSerializer.new(appointment).client_app_json
  end
end
```