

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
КАФЕДРА АВТОМАТИКИ ТА УПРАВЛІННЯ В ТЕХНІЧНИХ СИСТЕМАХ

«На правах рукопису»
УДК 004.4_____

«До захисту допущено»

Завідувач кафедри
_____ О. І. Ролік
(підпис) (ініціали, прізвище)

“ _____ ” _____ 20__ р.

Магістерська дисертація

зі спеціальності (спеціалізації) 121, інженерія програмного забезпечення
(інженерія програмного забезпечення комп'ютерних систем) _____
(код і назва спеціальності)

на тему: Система годування тварин на базі мінікомп'ютера _____

Виконав (-ла): студент (-ка) 2 курсу, групи ІТ-83мп
(шифр групи)

_____ Бойчук Роман Іванович _____
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник доцент кафедри АУТС, к.т.н, доцент Катін П.Ю. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____
(назва розділу) (науковий ступінь, вчене звання, прізвище, ініціали) (підпис)

Рецензент _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації
немає запозичень з праць інших авторів без
відповідних посилань.

Студент _____
(підпис)

Київ – 2019 року

рішення, реалізувати логіку пристрою, реалізувати серверну частину, розробити макет

6. Орієнтовний перелік ілюстративного (графічного) матеріалу Діаграма прецедентів системи, ER-діаграма, структурна діаграма пристрою, діаграма компонентів серверної частини, діаграма послідовності передачі команди

7. Орієнтовний перелік публікацій _____

8. Консультанти розділів дисертації^{1*}

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

9. Дата видачі завдання 5 вересня 2019

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
1	Порівняльний аналіз існуючих рішень	10.09.19 - 15.09.19	
2	Визначення основних вимог до системи	16.09.19 - 20.09.19	
3	Розроблення сценаріїв використання системи	21.09.19 - 28.09.19	
4	Вибір технологій для розробки	28.09.19 - 01.10.19	
5	Розроблення структурної схеми пристрою	02.10.19 - 07.10.19	
6	Розроблення ER-діаграми	08.10.19 - 14.10.19	
7	Реалізація бізнес-логіки	15.10.19 - 29.10.19	
8	Розроблення інтерфейсу користувача	30.11.19 - 03.11.19	
9	Створення макету	04.11.19 - 09.11.19	
10	Розроблення стартап-проекту	10.11.19 - 12.11.19	
11	Оформлення текстового матеріалу	13.11.19 - 20.11.19	
12	Оформлення графічного матеріалу	21.11.19 - 29.11.19	
13	Подача дисертації на перевірку	30.11.19 - 03.12.19	

Студент _____
(підпис)

Бойчук Р. І.
(ініціали, прізвище)

Науковий керівник дисертації _____
(підпис)

Катін П. Ю.
(ініціали, прізвище)

^{1*} Консультантом не може бути зазначено наукового керівника

РЕФЕРАТ

Магістерська дисертація містить 123 сторінки пояснювальної записки, 29 рисунків, 51 таблиці, 8 креслеників та 48 посилань на використані літературні джерела.

Актуальність даної роботи полягає в тому, що розглядає вирішення проблеми економії часу власників тварин та дозволяє організувати більш здорове харчування за розкладом. Використання Telegram бота як користувацького інтерфейсу системи відрізняє її від аналогів.

Мета дисертації – розробка автоматичної системи годування тварин з використанням мінікомп'ютера.

Об'єктом дослідження дисертації є робота вбудованих систем для годування тварин. Предметом дослідження є створення системи автоматичного годування тварин на базі мінікомп'ютера.

При вирішенні поставлених завдань застосовувалися загальні методи наукового пізнання. За допомогою них було досліджено існуючі аналоги, визначено основні шляхи для покращення.

Результати роботи можуть бути застосовані в реальному продукті – систему годування встановлено у корпус, під'єднано до механчних частин та випущено на ринок. Окремі компоненти системи, можуть бути використані для побудови ботів іншого спрямування чи створення іншої програмно-апаратної реалізації.

Ключові слова: система годування, мінікомп'ютер Raspberry Pi, сервоприводи, Telegram бот, Asp.Net Core.

SUMMARY

The master's thesis contains 123 pages of explanatory note, 29 drawings, 51 tables, 8 drawings and 48 references to used literature sources.

The relevance of this work is that it considers addressing the problem of saving time for animal owners and allows you to organize healthier meals on schedule. Using Telegram bot as the system user interface distinguishes it from its counterparts.

The purpose of the dissertation is to develop an automatic feeding system for animals using a minicomputer.

The object of the research is the work of embedded systems for animal feeding. The subject of the study is the creation of a system of automatic feeding of animals based on a minicomputer.

General methods of scientific cognition were used in solving the tasks. With them the existing counterparts have been identified and ways to improve have been found.

The results of the work can be applied to a real product - the feeding system is installed in the housing, connected to mechanical parts and released on the market. Individual components of the system can be used to build bots of other directions or to create other software and hardware implementations.

Keywords: feeding system, Raspberry Pi minicomputer, servos, Telegram bot, Asp.Net Core.

ЗМІСТ

ПЕРЕЛІК ТЕРМІНІВ І СКОРОЧЕНЬ	10
ВСТУП.....	11
1 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ.....	12
1.1 Огляд типів автоматичних годівниць	12
1.2 PetWant PF-103	14
1.3 Petkit Smart Feeder	15
1.4 Smart feed від Pet safe.....	16
1.5 Підсумки.....	17
2 ФОРМУВАННЯ ВИМОГ ДО СИСТЕМИ.....	18
2.1 Призначення системи	18
2.2 Складові системи.....	18
2.3 Вимоги до пристрою	19
2.3.1 Вимоги до режимів функціонування системи	19
2.3.2 Вимоги до функціональності при різних режимах зв'язку	20
2.3.3 Вимоги до годування	20
2.3.4 Вимоги до можливостей налаштування пристрою.....	20
2.4 Вимоги до серверної частини	21
2.4.1 Вимоги до зберігання даних	21
2.4.2 Вимоги до інтеграції з зовнішніми системами	22
2.5 Підсумок. Таблиця функціональних та нефункціональних вимог	22
3 СЦЕНАРІЇ ВИКОРИСТАННЯ СИСТЕМИ.....	24
3.1 Пристрій	24
3.2 Додаток.....	25

3.3 Зовнішня система	26
3.4 Web-interface	28
3.5 Опис прецедентів системи	28
4 ВИБІР ТА ОБГРУНТУВАННЯ ЕЛЕМЕНТІВ ТА ТЕХНОЛОГІЙ	45
4.1 Обґрунтування компонентів для пристрою.....	45
4.1.1 Вибір мінікомп'ютера.....	45
4.1.2 Використання сервоприводів.....	47
4.1.3 Вибір модулю керування сервоприводами	49
4.1.4 Вибір операційної системи для пристрою.....	50
4.1.5 Вибір мови програмування для пристрою.....	51
4.2 Обґрунтування вибору компонентів серверної частини.....	53
4.2.1 Вибір фреймворку для серверної частини.....	53
4.2.2 Вибір Nuget пакетів.....	54
5 БУДОВА СИСТЕМИ.....	56
5.1 Структурна схема підсистеми пристрою	56
5.1.1 Raspberry Pi	56
5.1.2 Модуль PCA9685 та сервоприводи	57
5.1.3 Кнопки та датчик закінчення корму	58
5.2 Діаграма компонентів серверної частини.....	58
6 РОЗРОБКА СХЕМИ БАЗИ ДАНИХ.....	61
6.1 Опис предметної області	61
6.2 Аналіз предметної області.....	62
6.3 Схема БД	63
7 РЕАЛІЗАЦІЯ БІЗНЕС-ЛОГІКИ СИСТЕМИ.....	69
7.1 Багатошарова архітектура	69

7.2 Паттерни, що застосовуються.....	74
7.2.1 Inversion of control	74
7.2.3 Command query responsibility segregation.....	75
7.3 Процес взаємодії підсистем	76
7.3.1 Процес взаємодії Telegram із сервером	76
7.3.2 Процес взаємодії серверної частини та пристрою.....	77
8 АЛГОРИТМ РОБОТИ ПРИСТРОЮ	79
9 РОЗРОБКА ІНТЕРФЕЙСУ КОРИСТУВАЧА.....	82
9.1 Початок роботи з ботом.....	82
9.2 Отримання довідки	83
9.3 Способи взаємодії з інтерфейсом	84
9.4 Функціональність команд.....	87
9.4.1 Встановити пристрій за замовчуванням	88
9.4.2 Графік на сьогодні.....	89
9.4.3 Модифікації графіку годування.....	90
9.4.4 Команда «Годувати зараз»	90
9.4.5 Встановлення порції	92
9.4.6 Перегляд статистики годувань	92
9.5 Макет пристрою	93
9.6 Висновки	94
10 СТАРТАП.....	96
10.1 Опис ідеї проекту	96
10.2 Технологічний аудит ідеї проекту.....	98
10.3 Аналіз ринкових можливостей запуску стартап-проекту.....	99
10.4 Розроблення ринкової стратегії проекту	108

10.5 Розроблення маркетингової програми стартап-проекту	110
10.6 Висновки до розділу	115
ВИСНОВКИ.....	117
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	120
ДОДАТОК А.....	Ошибка! Закладка не определена.
ДОДАТОК Б	Ошибка! Закладка не определена.
ДОДАТОК В.....	Ошибка! Закладка не определена.
ДОДАТОК Г	Ошибка! Закладка не определена.
ДОДАТОК Г	Ошибка! Закладка не определена.
ДОДАТОК Д.....	Ошибка! Закладка не определена.
ДОДАТОК Е	Ошибка! Закладка не определена.
ДОДАТОК Є	Ошибка! Закладка не определена.

ПЕРЕЛІК ТЕРМІНІВ І СКОРОЧЕНЬ

API – Application Programming Interface

HTTPS – Hypertext Transfer Protocol Secure

JSON – JavaScript Object Notation

OS – Operating System

MVC – Model View Controller

URL – Uniform Resource Locator

ASP – Active Server Pages

CQRS – Command Query Responsibility Segregation

IoC – Inversion of Control

Мессенджер – система миттєвого обміну повідомленнями через комп'ютерні мережі, призначена в першу чергу для обміну текстовими повідомленнями між людьми та між людьми і комп'ютерами

Бот – мається на увазі саме чат-бот, програма, що інтегрується з месенджером і взаємодія з якою ведеться як діалог

ВСТУП

Багато жителів міст мають домашніх тварин. Кожного дня ці мільйони людей спішать на роботу. А це означає, що тварини залишаються без догляду та нагляду близько 9-10 годин кожного дня.

Ця ситуація породжує ряд проблем. Однією з головних є те, що тварин недогодовують, або перегодовують насипаючи їм з лихвою повні миски їжі. Та це не завжди добре впливає на здоров'я домашньої тварини.

Проблема, яка постає для вирішення у даній роботі – годування домашніх тварин за відсутності людей. Де головним аспектом є акцент на здоровому харчуванні, при якому тварина не отримувала б дискомфорт у вигляді скаженого голодування протягом цілого дня, поки власники тварини зайняті на роботі, але і не переїдала, таким чином набираючи зайву вагу.

Сьогодні, в еру широкого розповсюдження комп'ютерів, інформаційних систем, та автоматизації процесів виробництва, багато проблем можна вирішити з допомогою пристроїв.

Останнім часом популярним способом застосування розумної електроніки для дому є використання мінікомп'ютерів. При малих масштабах вони мають достатньо потужну обчислювальну потужність, а наявність готових модулів-компонентів для апаратних систем, дозволяють достатньо швидко створювати розумні системи.

Тому пропонується вирішувати окреслену проблему за допомогою автоматизації процесу годування домашніх тварин, розробивши систему годування тварин на базі мінікомп'ютера. При цьому приділивши увагу розробці всіх складових системи. Як апаратній, програмній частині пристрою так і програмній частині серверу для даної екосистеми.

В останні роки серед загалу стали популярними месенджери такі як Telegram, Viber, WhatsApp. Месенджер Telegram користується попитом в Україні, тому в даному проекті розглядається також розробка інтерфейсу Telegram ботів для взаємодії з користувачами.

1 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

1.1 Огляд типів автоматичних годівниць

На ринку достатньо широко представлено годівниці для тварин різного рівня «розумності». Так за пошуком в Google «автоматична годівниця для тварин» спочатку потрапляєш на механічні годівниці, що просто нескінченно насипають сухий корм, доки не спорожниться контейнер з кормом. Та якщо пошукати англійською чи російською мовами, то вдасться знайти дійсно автоматичну годівницю для домашнього улюбленця.

Наявні на ринку автоматичні годівниці умовно можна поділити на 3 типи[1]: годівниці з відкидною кришкою, сегментні годівниці, програмовані. В свою чергу програмовані можна поділити на просто автоматичні, та розумні, що підключені до мережі.

Годівниці з відкидною кришкою представляють собою миски на одну порцію із таймером. Користувач налаштовує час о котрій миска відкриється. Мінусами такої системи є звичайно одноразовість та мала степінь автоматизації, а перевагами – менша порівняно з іншими класами ціна та малий розмір та можливість залишати їжу будь-якого типу.

Сегментовані годівниці, в свою чергу є логічним продовженням у розвитку попереднього типу. Кожен сегмент по суті – це одна годівниця з відкидною кришкою. Користувач так само як і для всіх інших типів налаштовує час прийому їжі. В залежності від виробника та ціни – таких прийомів 3-6, зрідка більше. Як виглядають годівниця із відкидною кришкою та сегментована зображено на рис. 2.1. Перевагою другого типу є більша кількість підгодовувань.



Рисунок 2.1 – Годівниця з відкидною кришкою (зліва) та сегментована (справа)

Характерною особливістю програмованих годівниць є наявність достатньо великого контейнеру, дозатору корму та можливості встановлення будь-якого графіку годування. Так, нажаль, всі програмовані годівниці не мають можливості використовувати щось інше ніж сухий корм. Але сучасні сухі корма є дуже збалансованою їжею та й використання саме сухого корму дозволяє дозувати порцію для тварини. З іншого боку вища ціна є недоліком для цього типу та більші розміри, у порівнянні з іншими. Проте перевагою для цього типу годівниць є велика кількість годувань, що дозволяє хазяїну домашньої тварини відлучитись на довший час. Репрезентабельними прикладами автоматичних програмованих годівниць є модель «Healthy Pet Simply Feed» від компанії PetSafe (Рис. 2.2). Габаритна модель для хорошого запасу корму із невеликим дисплеєм як у калькулятора та декількома кнопками для налаштування часу та порції.

Серед програмованого типу автоматичних годівниць варто виділити розумні автоматичні годівниці. Характерно їх відрізняє підключеність до мережі, зазвичай Інтернету. Це відкриває можливості віддаленого управління.



Рисунок 2.2 – Автоматична програмована годівниця «Healthy Pet Simply Feed» від компанії PetSafe [2]

Оскільки проект розглядає побудову саме сучасної системи годування тварин, то за аналоги варто взяти саме розумні автоматичні годівниці. Яскравими прикладами таких годівниць виступають PetWant PF-103[3], Petkit Smart Feeder[4], Smart feed від Pet safe[5].

1.2 PetWant PF-103

Головною задачею PetWant PF-103 є порційне годування домашньої тварини за розкладом. Це дозволяє запобігти переїданню та подовжує життя домашніх тварин. На рис. 2.3 зображено даний пристрій.

Основними особливостями даного пристрою є:

- автономний режим (батареї задіяні, якщо вдома раптом вимкнулась подача електроенергії);
- порційне годування з можливістю заданого розміру порції (мінімальна порція – 10г);
- вбудована камера, що дозволяє робити фото, відео;



Рисунок 2.3 – Розумна годівниця PetWant PF-103 [3]

- можливість відео-аудіо зв'язку із твариною, що дозволяє знизити рівень стресу у домашньої тварини та посварити у випадку, якщо вона коїть шкоду;
- управління пристроєм із смартфона;
- нотифікація на телефон у разі, якщо корм закінчується;
- додаток для Android та iOS.

1.3 Petkit Smart Feeder

Виробник пристрою Petkit Smart Feeder позиціонує його як надійну годівницю, корм в якій ніколи не застряє. Корм зберігається в підходящих для нього сухих умовах та герметично. На рис. 2.4 зображено даний пристрій.



Рисунок 2.3 – Розумна годівниця Petkit Smart Feeder [4]

Основними особливостями даного пристрою є:

- надійність системи подачі/дозування корму;
- можливість налаштування плану годування або можливість погодувати прямо зараз через мобільний додаток;
- мобільний додаток для налаштування пристрою для Android та iOS;
- відсутність камери та мікрофона, що задовольняє вимоги деяких людей, для яких приватність є надважливою;
- акумулятор на випадок аварійної ситуації з живленням;
- можливість управління декількома годівницями з одного пристрою.

1.4 Smart feed від Pet safe

Автоматична годівниця для домашніх тварин PetSafe Smart Feed - це зручне рішення, яке дозволяє миттєво годувати домашнього улюбленця під час роботи, виконуючи доручення чи під час великої зайнятості по дому (особливо актуально може бути, якщо дім великий). Годівниця підключається до домашнього бездротового маршрутизатора. Вміщує 24 стакани корму, та дозволяє забезпечує графік годування до 12 разів на день.

Особливостями даного пристрою є:

- можливість налаштування різного розміру порції на різні прийоми їжі;
- запас батареї на випадок відсутності електропостачання;
- функція повільного годування, що корисна якщо тварина переїдає постійно; дана функція видаватиме пайок не одразу а протягом п'ятнадцяти хвилин;
- додаток для налаштування пристрою для платформ iOS та Android;
- під'єднання до Amazon Echo[6], що дає можливість через голосового асистента Amazon Alexa[7] покормити домашню тварину;
- повідомлення користувачеві про відсутність зв'язку з пристроєм, про те, що

- корм закінчився, скоро закінчиться, що улюбленець покормлений;
- автоматичне замовлення на покупку корму, коли він закінчується.

1.5 Підсумки

На підставі огляду існуючих розумних систем годування тварин, можна виокремити наступні особливості, що притаманні сучасним пристроям даного класу. Саме ці особливості можуть бути покладені в основу розробки магістерського проекту. Серед них:

- достатньо місткий контейнер, що дозволяє користувачу довгий час не згадувати про насипання корму;
- гнучкі можливості налаштування графіку годування та розмірів порцій;
- повідомлення користувачеві про статус годівниці, наприклад, закінчення корму у контейнері, втрата зв'язку із пристроєм;
- додаток для мобільного пристрою, що дасть змогу не встановлювати зайві елементи керування у годівницю та дає можливість віддаленого керування;
- можливості керувати декількома годівницями з одного мобільного пристрою, та можливість керувати однією годівницею з декількох мобільних пристроїв.

Окремо варто згадати наявність динаміку, камери. Представлені моделі відрізняються наявністю цих елементів. Можливо саме через те, що для багатьох людей приватність на першому місці, камери не встановлені на кожному пристрої. Але загалом дбати про тварин, при цьому підтримуючи з ними зв'язок, використовуючи розумні системи годування стало легше.

Все ж є деякі функції, що можуть мати попит та досі не представлені на ринку. Деякі тварини не завжди правильно харчуються, що погіршує їх здоров'я та може призвести до хвороб органів травлення[8], тому важливо слідкувати за їх дієтою. Друга ж функція, яка могла б трохи спростити життя користувачів, це можливість керування через месенджери, такі як Telegram[9] та Viber[10].

2 ФОРМУВАННЯ ВИМОГ ДО СИСТЕМИ

2.1 Призначення системи

Проект має на меті спростити та автоматизувати процес годування домашніх тварин, при цьому надати користувачеві можливість модерувати калорії домашньої тварини та віддалений доступ. Зокрема, дати користувачеві доступ до звітної інформації про проведені годування та заплановані.

Для цього система має бути спроможна виконувати такі задачі:

- забезпечення автоматизації процесу годування тварини;
- зберігання даних графіку годування, користувача, журналу годування;
- забезпечення можливості редагування, видалення інформації;
- забезпечення віддаленого інформування користувача;
- забезпечення віддаленого керування пристроєм;
- запобігання помилкам з боку користувача під час виклику команд;
- забезпечення захищеного зв'язку пристроїв з системою;
- забезпечення можливості інтеграції із зовнішніми системами;
- забезпечення повного контролю користувача над пристроєм.

2.2 Складові системи

Враховуючи завдання, що повинна виконувати система годування тварин, необхідно виділити 2 підсистеми:

- пристрій;
- сервер.

Пристрій – за задумкою стартапу, власне річ, яку може купити користувач, та поставити вдома. Це головний виконавчий механізм системи. Головними задачами пристрою є можливості:

- зберігання корму;
- видача корму за графіком;
- видача корму при надсиланні команди;

- сигналювання про закінчення корму;
- фотофіксація.

Сервер – за задумкою стартапу, знаходиться у володінні компанії. Кожен підключений до мережі Інтернет пристрій може зв'язуватись із сервером. Головна можливість серверної підсистеми:

- зберігання даних;
- видача даних за запитом;
- зв'язок з користувачем;
- передача команд між інтерфейсом користувача та сервером;
- передача даних між сервером та пристроєм;
- генерація та передача сповіщень-попереджень;

2.3 Вимоги до пристрою

2.3.1 Вимоги до режимів функціонування системи

Пристрій, як підсистема що передбачена для встановлення вдома, повинен вміти обробляти різні стани комунікації суміжних з ним пристроїв. Таким чином він повинен вміти працювати у різних режимах:

- наявний зв'язок з мережею Інтернет
- наявний зв'язок з сервером
- дані бездротової мережі не встановлені
- відсутній зв'язок із сервером

Варто зауважити, що режими можуть змінювати одне одного, тому однією із задач підсистеми пристрою є забезпечення обробки помилок та підтримки роботи у будь-яких режимах, незалежно від того, який з режимів був попереднім. Дані вимоги ляжуть в основу розробки можливих станів у яких може перебувати підсистема пристрою.

Зокрема, незважаючи на режими, в яких відсутній зв'язок із сервером, пристрій все одно повинен продовжувати роботу за графіком, що був збережений на ньому в режимі наявності зв'язку із сервером.

2.3.2 Вимоги до функціональності при різних режимах зв'язку

Припускається, що у режимі відсутності зв'язку прилад перебуватиме одразу після покупки під час першого запуску. Також у випадку непередбачуваних неполадок із мережею.

У цьому режимі мають бути забезпечені базові можливості:

- годування по кнопці
- увімкнення налаштування
- вимкнення\увімкнення пристрою

Варто зазначити, що годування по кнопці повинне бути реалізоване з урахуванням випадкових натискань, що можуть бути здійснені твариною, чи людиною. І таким чином кнопка повинна бути захищена від тварин.

У разі тривалої відсутності зв'язку, може накопичитись критичний об'єм інформації на пристрої. Необхідно передбачити обробку такого роду ситуацій, при цьому не створивши перешкод для подальшої роботи пристрою.

Пристрій, що має безперешкодну можливість з'єднуватись до сервера, повинен вчасно отримувати оновлення графіку (розкладу) автоматичного годування. А в разі відсутності продовжити годування за раніше отриманим графіком.

2.3.3 Вимоги до годування

Пристрій повинен мати можливість видачі порції корму певного розміру. Ця можливість є актуальною для випадків, коли необхідно обмежити споживання домашньої тварини, чи встановити для неї дієту. При цьому варто зберігати дані годування час та порція. Це дасть змогу побачити статистику з'їденого. Дані годування необхідно відправляти на сервер.

2.3.4 Вимоги до можливостей налаштування пристрою

Для зменшення елементів користувацького інтерфейсу на пристрої,

пропонується винести всю логіку по налаштуванню на мобільний телефон. Це з одного боку обмежить коло потенційних покупців, але з іншого дасть можливість створити більш функціональний та зручний інтерфейс керування пристроєм, а також дасть змогу імплементувати віддалене керування.

Можлива ситуація така, що пристрій продається, дарується, чи переходить до іншого користувача іншим шляхом. В цьому випадку важливо якимось чином передати можливість керування пристроєм від одного користувача іншому. Оскільки не завжди попередній власник пристрою доступний, доцільно тримати цю функціональність на самому пристрої. Отже варто передбачити кнопку скидання налаштувань, що дасть змогу новому власнику розірвати реєстрацію пристрою із попереднім власником.

Можлива і така ситуація, що пристрій куплений, увімкнений, але потрібно налаштувати зв'язок з Інтернет. Варто забезпечити цю можливість, урахувавши, що інтерфейс користувача має бути в смартфоні.

2.4 Вимоги до серверної частини

Сервер має забезпечити можливість користувачам взаємодіяти із своїми пристроями через систему «Telegram». Користувачі, відкривши бот в Telegram мають використовувати його як інтерфейс управління їхніми пристроями.

2.4.1 Вимоги до зберігання даних

Збережені дані повинні відповідати принципам несуперечності, цілісності, правдивості.

Підсистема серверу повинна зберігати наступні дані:

- відомості та інформація про користувача;
- відомості про пристрій;
- журнал годувань;
- графіки годування.

2.4.2 Вимоги до інтеграції з зовнішніми системами

Було б зручно скористатися можливостями сторонніх систем таких як месенджери. Наприклад Telegram, Viber мають можливості по взаємодії з користувачами з допомогою ботів[11][12]. У розрізі даного проекту необхідно закласти можливість для майбутньої інтеграції з іншими системами. А як приклад реалізувати взаємодію користувача через Telegram.

2.4.3 Вимоги до інтерфейсу користувача

Користувач повинен мати повний контроль над налаштуванням графіку годування. В ідеалі таких графіків може бути декілька, а користувач може зробити один з них активним. Має забезпечуватись змога користувача додавати до розкладу певну, адекватну, кількість запланованих на день годувань. Користувач має мати змогу побачити заплановані годування та відмінити деякі з них у разі необхідності. Має бути реалізована команда перегляду графіку годування на сьогодні. Та команда годування зараз, що по ідеї є аналогом кнопки на пристрої. Сповіщення про певні стани пристрою чи попередження також повинні з'являтися на інтерфейсі користувача. Наприклад повинна бути можливість сповіщати користувача про закінчення корму чи тривалу відсутність зв'язку між сервером та пристроєм.

2.5 Підсумок. Таблиця функціональних та нефункціональних вимог

Таблиця 2.1 - Нефункціональні вимоги

Підсистема	Вимога
Пристрій	Захищений зв'язок з сервером
	Відносно мала затримка доставки команд
	Надійність тривалого використання. Обробка будь-яких помилок
Сервер	Зручність та зрозумілість інтерфейсу користувача

	Якість роботи
	Захищені комунікації

Таблиця 2.2 - Функціональні вимоги

Підсистема	Вимога
Пристрій	Автоматична видача корму по графіку
	Визначення ситуації закінчення корму, сигналювання про неї на сервер
	Ручне годування по кнопці на пристрої
	Кнопка скидання налаштувань
	У режимах відсутності зв'язку із сервером, продовжити годування за раніше отриманим графіком
	У режимі наявності зв'язку, отримувати оновлення поточного графіку годування та отримувати команди
Сервер	Зберігання даних графіку запланованих годувань, журналу проведених годувань, інформації користувачів та інших необхідних для функціонування системи даних
	Інтеграція із системою Telegram, інтерфейс користувача використовуючи можливості ботів
	Можливості перегляду \ створення \ редагування \ видалення графіків годування
	Команди «Годувати зараз» та «Вимкнути»
	Генерація сповіщень, попереджень. Наприклад про тривалу відсутність зв'язку з пристроєм.
	Забезпечення передачі інформації між пристроєм та сервером, сервером та додатком.

3 СЦЕНАРІЇ ВИКОРИСТАННЯ СИСТЕМИ

Система годування тварин передбачає апаратну і програмну частини. Отже буде пристрій і певний інтерфейс користувача. Інтерфейс користувача це не одна підсистема, а декілька. Тому можна виділити способи взаємодії із системою: фізична взаємодія із пристроєм, що стоятиме у користувача вдома, взаємодія користувача через інтерфейс зовнішньої системи (Telegram, Viber тощо), взаємодія користувача через мобільний додаток, взаємодія адміністратора компанії через веб інтерфейс.

Виходячи з типів взаємодії користувача можна виділити 2 актори:

- користувач;
- адміністратор.

Та виділити 4 підсистеми:

- пристрій;
- зовнішня система (Telegram, Viber або подібна);
- додаток;
- web-interface.

3.1 Пристрій

Кінцевий користувач отримає пристрій із декількома кнопками та без екрану. Більшість маніпуляцій проводитиметься з віддаленого інтерфейсу (месенджеру). Та декілька кнопок все ж потрібні: «увімкнути/вимкнути», «ручного годування», «увімкнення доналаштування», «скидання всіх налаштувань». Доцільно об'єднати деякі з кнопок для економії компонентів.

Запуск відбуватиметься перемикачем «увімкнути вимкнути». Під час увімкнення пристрою та якщо пристрій працюватиме від мережі, автоматично увімкнеться режим доналаштування (Bluetooth), що дасть змогу підключитись із додатку та передати дані підключення до бездротової мережі маршрутизатора wifi. Цей режим за замовчуванням буде працювати певну кількість хвилин, а згодом вимикатись.

Можлива ситуація коли пристрій перейде до іншого користувача, тобто стане необхідність переналаштування системи годування. Тоді кнопка «скидання всіх налаштувань» стане у пригоді.

Доналаштування повинне вмикатись автоматично, при увімкненні пристрою. Може статись так, що користувач вперше увімкнувши пристрій, не зареєструється і не встановить дані для під'єднання до локальної мережі. Тоді ця функціональність стане у пригоді. Доцільним є також, якщо додати функцію увімкнення режиму доналаштування на кнопку «скидання всіх налаштувань» таким чином, що утримавши кнопку 5 с – увімкнеться доналаштування, а утримавши 20 – скидання всіх налаштувань.

Кнопка "Годувати зараз" для ручного годування має бути зручно розташована, але при цьому захищеною від випадкових натискань твариною. Вона потрібна на випадок економії часу, або необхідності екстреного годування. Наприклад, якщо потрібно кудись йти разом із твариною – можливо натиснути дану кнопку завчасно, до запланованого годування, та, погодувавши тварину, рушити. Для підтвердження намірів годування планується ввести мінімальний час (наприклад 5с) який треба утримувати кнопку.

Підсумовуючи, можна виділити такі сценарії використання пристрою користувачем:

- увімкнення / Вимкнення;
- скидання всіх налаштувань;
- увімкнення доналаштування;
- ручне годування.

3.2 Додаток

Необхідність окремого додатку в тому, щоб забезпечити даними мережі пристрій та зареєструвати користувача. Вказати ідентифікатор та пароль домашньої мережі, таким чином організувавши зв'язок пристрою з сервером. Вказати ім'я та нікнейм для підключення до зовнішньої системи (Telegram). Також, якщо впроваджувати даний

проект в життя, доцільним може виявитись реалізація всього необхідного функціоналу для налаштування графіку годування у додатку.

При первинному налаштуванні пристрою за допомогою додатку, можна налаштувати інтеграцію з одною із зовнішніх систем (месенджерів). Вказавши свій нікнейм, необхідно обрати для якого саме месенджеру цей нікнейм, та вказати пріоритет месенджерів. У випадку надсилання сповіщень пріоритет дасть змогу системі спробувати надіслати в систему з першим пріоритетом, і, у випадку невдачі, у другу і так далі.

Необхідність додатку виражена у наступних сценаріях використання:

- реєстрація користувача;
- під'єднання до пристрою;
- реєстрація пристрою;
- налаштування до бездротової мережі;
- вибір системи інтеграції (месенджеру).

Необхідно відзначити, що перший сценарій включає в себе наступні.

3.3 Зовнішня система

Сьогодні достатньо просто можна інтегруватись із месенджерами, що надають Bot API, такі як Telegram[11], Viber[12], що мають провідні положення на українському ринку[13]. В рамках даного проекту розкривається побудова лише інтеграції з Telegram. Але сценарії використання, за задумом, лишаться такими ж для всіх систем, для яких буде реалізована інтеграція. Надалі дані сценарії використання через інтерфейс зовнішньої системи.

Основною функцією пристрою є автоматизація годування домашніх тварин. А така автоматизація потребує графіку годування.

Пересічному користувачу, має бути досить розпорядку годування декількох типів: щоденне, обраний день тижня, на дату. Можливо колись доведеться забезпечити інші типи, тому важливо залишити для цього можливість. Створення графіку має передбачати введення декількох рядків різного типу розпорядку із

вказанням часу. Тобто в рамках одного графіку може бути вказано і щоденний розпорядок годування і на конкретну дату.

Різні типи розпорядку мають різну вагу, так якщо в межах графіку заплановано щось саме на певну дату, то будуть використовуватись лише рядки для певної дати, а щоденний розпорядок та розпорядок по дням тижня ігноруватиметься.

В свою чергу потенційній компанії виробнику даного пристрою це дасть можливість створити графік годування – дієту. А користувач у свою чергу зможе обрати певну дієту, вказавши параметри, після чого дієта буде застосована для його пристрою.

Переглядаючи план годування на сьогодні, користувач може відмінити певні заплановані годування.

Також важливо, щоб користувач мав змогу переглянути статистику (журнал) годування: скільки сьогодні разів було погодовано, скільки на цьому тижні, побачити пропуски в годуваннях.

Користувач, що зареєстрував пристрій має права керуючого пристрою. Він може дати доступ до налаштування графіків іншим користувачам. Таким чином створиться коло людей, які мають доступ до цього пристрою. Загалом, один користувач може мати доступ, чи доступ керування до багатьох пристроїв.

Важливо, щоб вся система працювала справно, і в разі певних аномалій повідомляла користувача. Тому сповіщення (попередження) про можливі негаразди підлягають реалізації. Приклади таких сповіщень: якщо корм закінчується, якщо тварина не поїла, якщо пристрій певний час вже немає зв'язку сервера системи з пристроєм, якщо не вдалося надіслати новий графік годування на пристрій.

Підсумовуючи, можна виділити такі основні сценарії використання:

- обрати існуючий графік дієти;
- створити графік;
- налаштувати графік годування;
- переглянути план на сьогодні;
- отримати дані журналу годування;
- дати доступ іншим користувачам;

- отримати попередження.

3.4 Web-interface

Адміністратор із стороною компанії повинен мати змогу ввести в систему дані про виготовлені пристрої, які надалі можуть бути використані для валідації з'єднань із сервером. Інтерфейс для адміністратора краще всього зробити у вигляді веб-інтерфейсу.

Створення шаблонів графіків годування чи графіків дієти також повинне бути адміністратору на початкових етапах. В майбутньому потенційно можливо автоматизоване створення графіків з допомогою штучного інтелекту.

Призначення веб-інтерфейсу – для адміністратора з компанії, а сценарії використання:

- внесення пристрою в систему;
- створення графіку дієти;
- керування інтеграцією з системами.

3.5 Опис прецедентів системи

Таблиця 3.1 - Опис прецеденту увімкнення пристрою

Назва прецеденту	Увімкнути пристрій
Підсистема	Пристрій
Унікальний ідентифікатор	П_001
Опис	Користувач може увімкнути куплений пристрій
Актори	Користувач
Бізнес значимість	Необхідна умова для інших сценаріїв
Частота використання	Рідко. В ідеалі користувач одного разу вмикає пристрій.
Тригери	Перемкнути перемикач вимкн / увімкн.

Передумови	Пристрій вимкнений
Післяумови	Пристрій увімкнений
Виключення	Якщо є живлення відсутнє, пристрій залишиться вимкненим

Таблиця 3.2 - Опис прецеденту вимкнення пристрою

Назва прецеденту	Вимкнути пристрій
Підсистема	Пристрій
Унікальний ідентифікатор	П_002
Опис	Користувач може вимкнути увімкнений пристрій
Актори	Користувач
Бізнес значимість	Необхідна функціональність, задоволення користувача
Частота використання	Рідко. В ідеалі користувач одного разу вмикає пристрій і не вимикає довгий час.
Тригери	Перемкнути перемикач увімкн / вимкн.
Передумови	Пристрій увімкнений
Післяумови	Пристрій вимкнений
Виключення	Неможливі

Таблиця 3.3 - Опис прецеденту скидання всіх налаштувань пристрою

Назва прецеденту	Скидання всіх налаштувань
Підсистема	Пристрій
Унікальний ідентифікатор	П_003
Опис	Користувач має можливість скинути всі налаштування пристрою та графіків годування.
Актори	Користувач
Бізнес значимість	Можливість продати повернутий пристрій, якщо він вже був у користуванні. Передача пристрою користувачем іншому користувачу.
Частота використання	Рідко. Користувач в переважній більшості випадків налаштовує пристрій одного разу.
Тригери	Утримування кнопки скидання 20 с
Передумови	Пристрій увімкнений (П_001)
Післяумови	Всі налаштування пристрою і пов'язані облікові записи користувачів видаляються.
Основний шлях	Натиснути кнопку скидання Утримувати 20 с
Виключення	Непередбачені

Таблиця 3.4 - Опис прецеденту увімкнення доналаштування

Назва прецеденту	Увімкнення доналаштування
Підсистема	Пристрій
Унікальний ідентифікатор	П_004
Опис	Користувач може увімкнути Bluetooth для доналаштування бездротової мережі з допомогою смартфона
Актори	Користувач
Бізнес значимість	Увімкнення
Частота використання	Рідко. Налаштування бездротової мережі необхідне лише одного разу та у випадку зміни даних мережі
Тригери	Утримання кнопки доналаштування протягом 5 с
Передумови	Пристрій увімкнений (П_001)
Післяумови	Увімкнений Bluetooth на пристрої. Мигання синім кольором індикатора.
Основний шлях	Натиснути кнопку скидання Утримувати 5 с
Виключення	Bluetooth не увімкнувся: 1. Пристрій повертається у попередній стан 2. Вимикає/вмикає пристрій 3. Знову повторює дії основного шляху

Таблиця 3.5 - Опис прецеденту ручного годування

Назва прецеденту	Ручне годування
Підсистема	Пристрій
Унікальний ідентифікатор	П_005
Опис	Користувач може запустити годування в будь-який момент часу
Актори	Користувач, Сервер
Бізнес значимість	Необхідність такого функціоналу у разі відсутності налаштувань мережі, задоволення потреб користувача
Частота використання	Більшість користувачів не будуть використовувати
Тригери	Користувач утримує кнопку «ручного годування» протягом 4с
Передумови	Пристрій увімкнений (П_001), не перебуває у стані годування в момент утримання кнопки
Післяумови	Перехід пристрою у стан годування. Або індикація червоним кольором у разі виключення. Виключення: неможливо годувати якщо режим суворої дієти, відсутність корму
Основний шлях	Натиснути кнопку «ручного годування» Утримувати 4 с
Альтернативні шляхи	Надіслати команду годування із зовнішньої системи (месенджеру)
Виключення	Режим суворої дієти: Вимкнення режиму суворої дієти Відсутність корму: насипати корм у контейнер знову спробувати дії основного чи альтернативних шляхів

Таблиця 3.6 - Опис прецеденту під'єднання до пристрою

Назва прецеденту	Під'єднання до пристрою
Підсистема	Додаток
Унікальний ідентифікатор	П_006
Опис	У додатку користувач має обрати пристрій, до якого він хоче під'єднатись
Актори	Користувач
Тригери	У списку знайдених додатком пристроїв обрати пристрій для приєднання
Передумови	На пристрої увімкнений Bluetooth (П_004), користувач відкрив додаток на смартфоні
Післяумови	Перехід пристрою у стан налаштування, перехід до інтерфейсу налаштування у додатку
Основний шлях	Відкрити додаток, натиснути пошук пристроїв, обрати пристрій
Виключення	Пристрої не знайдено: Повідомити користувача

Таблиця 3.7 - Опис прецеденту реєстрації користувача

Назва прецеденту	Реєстрація користувача
Підсистема	Додаток
Унікальний ідентифікатор	П_007
Опис	Користувач має зареєструвати себе, пристрій та обрати зовнішню систему інтеграції
Актори	Користувач
Бізнес значимість	Забезпечення доступу користувачу із зовнішньої системи, задоволення користувача
Частота використання	Рідко. Користувач повинен зробити це хоча б один раз
Тригери	Ввести дані у форму, підтвердити

Передумови	П_006
Післяумови	Перехід пристрою у зареєстрований стан. Користувач тепер може використовувати зовнішню систему інтеграції та сценарії використання зовнішньої системи Виключення: див. виключення
Основний шлях	П_006 Ввести дані у форму Підтвердити
Альтернативні шляхи	Надіслати команду годування із зовнішньої системи (месенджеру)
Виключення	Не вдалося зареєструвати користувача: повідомити користувача, пристрій залишається незареєстрованим

Таблиця 3.8 - Опис прецеденту створення графіку

Назва прецеденту	Створити графік
Підсистема	Зовнішня система
Унікальний ідентифікатор	П_008
Опис	Користувач може створити графік годування
Актори	Користувач
Тригери	Користувач з допомогою інтерфейсу зовнішньої системи створює порожній графік
Передумови	Користувач зареєстрований (П_007), Пристрій увімкнено (П_001)
Післяумови	Створено запис графіку у БД
Основний шлях	Зайти в месенджер, натиснути кнопку створити графік у бота
Виключення	Не вдалось створити: Повідомити користувача

Таблиця 3.9 - Опис прецеденту додавання рядку часу

Назва прецеденту	Додати рядок часу
Підсистема	Зовнішня система
Унікальний ідентифікатор	П_009
Опис	Користувач може додати рядок часу до графіку годування
Актори	Користувач
Тригери	Користувач з допомогою інтерфейсу зовнішньої системи вводить час годування та зберігає рядок часу
Передумови	Створено графік (П_008) до якого належатиме даний рядок
Післяумови	Створено запис рядку часу у БД

Основний шлях	Зайти в месенджер, ввести команду та час для створення рядку часу (годування)
Виключення	Неправильно сформована команда: Повідомити користувача

Таблиця 3.10 - Опис прецеденту видалення рядку часу

Назва прецеденту	Видалити рядок часу
Підсистема	Зовнішня система
Унікальний ідентифікатор	П_010
Опис	Користувач може видалити рядок часу
Актори	Користувач
Бізнес значимість	Забезпечення основних сценаріїв використання системи
Частота використання	Декілька разів, після створення графіку. Користувач повинен налаштувати графік годування, додавши рядки часу (годування), та за необхідності видаливши непотрібні
Тригери	Введення команди видалення у зовнішній системі,
Передумови	Користувач переглядає створені рядки часу (П_010)
Післяумови	Видалено запис у БД
Основний шлях	Запросити графік годування Натиснути видалення рядку часу Підтвердити
Альтернативні шляхи	Видалити графік годування
Виключення	Не вдалося здійснити операцію: повідомити користувача

Таблиця 3.11 - Опис прецеденту перегляду плану годування на сьогодні

Назва прецеденту	Переглянути план на сьогодні
Підсистема	Зовнішня система
Унікальний ідентифікатор	П_011
Опис	Користувач може переглянути заплановані на сьогодні годування.
Актори	Користувач
Бізнес значимість	Забезпечення основних сценаріїв використання системи
Частота використання	У випадку складних налаштувань графіку, ця функція буде корисною користувачеві
Тригери	Введення команди/кнопки перегляду плану на сьогодні
Передумови	Користувач має бути зареєстрованим (П_007), або мати доступ до пристрою (П_012)
Післяумови	Користувач бачить заповнений, або пустий графік годування
Основний шлях	Відкрити перегляд плану на сьогодні
Виключення	Не вдалося здійснити операцію: повідомити користувача

Таблиця 3.12 - Опис прецеденту надання доступу іншому користувачу

Назва прецеденту	Дати доступ іншому користувачеві
Підсистема	Зовнішня система
Унікальний ідентифікатор	П_012
Опис	Користувач може дати доступ іншим користувачам до пристрою
Актори	Користувач
Бізнес значимість	Забезпечення основних сценаріїв використання системи
Частота використання	У випадку, коли декілька користувачів хочуть мати можливість налаштування графіку
Тригери	Введення команди на надання доступу, введення даних (нікнейму) іншого користувача
Передумови	Користувач зареєстрований (П_007)
Післяумови	Додано запис у БД
Основний шлях	Ввести команду та дані іншого користувача
Виключення	Не вдалося знайти вказаного користувача: повідомити користувача, що вводив команду

Таблиця 3.13 - Опис прецеденту отримання даних журналу годування

Назва прецеденту	Отримати дані журналу годування
Підсистема	Зовнішня система
Унікальний ідентифікатор	П_013
Опис	Користувач може переглянути скільки разів того чи іншого дня відбувалось годування, і який розмір порції був при годуванні
Актори	Користувач
Бізнес значимість	Компанія, має змогу зібрати дані для аналізу і в подальшому проектувати наступну версію пристрою з урахуванням зібраних журналів
Частота використання	За необхідності
Тригери	Введення/обрання команди на перегляд журналу
Передумови	Користувач зареєстрований (П_007), або має доступ (П_012)
Післяумови	Користувачу відображено журнал годування (дані витягнуті із БД)
Основний шлях	Запросити журнал годування
Виключення	Не вдалося здійснити операцію: повідомити користувача Дані годування відсутні: повідомити користувача

Таблиця 3.14 - Опис прецеденту отримання попередження

Назва прецеденту	Отримати попередження
Підсистема	Зовнішня система
Унікальний ідентифікатор	П_014
Опис	Користувач може отримати повідомлення про певні стани системи. Наприклад попередження про закінчення корму чи про те що тварина не їсть
Актори	Користувач
Бізнес значимість	Забезпечення користувача необхідною інформацією, та наприклад використання певної реклами про корм, що може дати компанії додатковий прибуток
Частота використання	Часто. Корм, наприклад, закінчуватиметься щомісяця, а можливо й частіше – залежить від розміру порції
Тригери	Пристрій надсилає на сервер – сповіщення, яке згодом показується користувачу у зовнішній системі
Передумови	Користувач зареєстрований (П_007), або має доступ (П_012), а пристрій підключений до мережі
Післяумови	Відсутні
Основний шлях	Відкрити інтерфейс зовнішньої системи Переглянути сповіщення
Виключення	Відсутні

Таблиця 3.15 - Опис прецеденту встановити сигнал годування

Назва прецеденту	Встановити сигнал годування
Підсистема	Зовнішня система
Унікальний ідентифікатор	П_015
Опис	Користувач може встановити певний звуковий файл, як сигнал, що буде програватись перед видачею порції
Актори	Користувач
Бізнес значимість	Додаткова функціональність
Частота використання	Користувач матиме змогу змінити сигнал в будь-який момент
Тригери	Введення команди видалення у зовнішній системі,
Передумови	Користувач зареєстрований (П_007)
Післяумови	Сигнал збережено на сервері для передачі на пристрій (див. Виключення)
Основний шлях	Надіслати аудіо файл через інтерфейс зовнішньої системи
Альтернативні шляхи	-
Виключення	Не вдалося зв'язатись із пристроєм: повідомити користувача

Таблиця 3.16 - Опис прецеденту отримати знімок з пристрою

Назва прецеденту	Отримати знімок з пристрою
Підсистема	Зовнішня система
Унікальний ідентифікатор	П_016
Опис	Користувач може отримати знімок пристрою
Актори	Користувач
Бізнес значимість	Дозволяє перевірити власнику тварини, що його тварина в даний момент біля пристрою, та наприклад їсть корм
Частота використання	Декілька разів, після створення графіку. Користувач повинен налаштувати графік годування, додавши рядки часу (годування), та за необхідності видаливши непотрібні
Тригери	Введення команди видалення у зовнішній системі,
Передумови	Користувач переглядає створені рядки часу (П_010)
Післяумови	Видалено запис у БД
Основний шлях	Запросити графік годування Натиснути видалення рядку часу Підтвердити
Альтернативні шляхи	Видалити графік годування
Виключення	Не вдалося здійснити операцію: повідомити користувача

Таблиця 3.17 - Опис прецеденту внесення даних типу пристрою в систему

Назва прецеденту	Внесення даних типу пристрою в систему
Підсистема	Web-interface
Унікальний ідентифікатор	П_017
Опис	Адміністратор може внести дані нової типу (моделі) пристрою в систему

Актори	Адміністратор
Бізнес значимість	При випуску нових моделей, дані про них повинні бути в БД
Частота використання	По мірі випуску нових типів годівниць
Тригери	Введення інформації про новий тип
Передумови	Наявність доступу до системи
Післяумови	Створено записи у БД
Основний шлях	Відкрити веб-сторінку, ввести дані
Альтернативні шляхи	-
Виключення	Не вдалося здійснити операцію: повідомити адміністратора

Таблиця 3.18 - Опис прецеденту налаштування графіку дієти

Назва прецеденту	Налаштування графіку дієти
Підсистема	Web-interface
Унікальний ідентифікатор	П_018
Опис	Адміністратор може створити, змінювати, видалити графік дієти. Який користувачі зможуть скопіювати для свого пристрою.
Актори	Адміністратор
Бізнес значимість	Важлива для системи функція, що забезпечує здорове харчування тварин
Частота використання	Відносно рідко
Тригери	Внесення та зміна рядків часу для графіку дієти
Передумови	Наявність доступу до системи
Післяумови	Створено записи у БД
Основний шлях	Відкрити веб-сторінку, ввести дані
Альтернативні шляхи	-
Виключення	Не вдалося здійснити операцію: повідомити

	адміністратора
--	----------------

Всі вищеописані сценарії використання зображені на діаграмі сценаріїв використання (додаток X).

4 ВИБІР ТА ОБГРУНТУВАННЯ ЕЛЕМЕНТІВ ТА ТЕХНОЛОГІЙ

Для того, щоб почати розробку системи, необхідно розглянути існуючі компоненти, можливості для реалізації та обрати підходящі відповідно до вимог. Для даної системи необхідно обрати елементи та технології для реалізації серверної частини та пристрою.

4.1 Обґрунтування компонентів для пристрою

4.1.1 Вибір мінікомп'ютера

На ринку представлені багато різних моделей мінікомп'ютерів. Серед них BeagleBoard, Raspberry pi, Orange pi та плати, що теоретично могли б справитись із завданням даної системи такі як сімейство Stm32 discovery, Arduino.

Beagle board — одноплатний комп'ютер, розроблений компаніями Texas Instruments і Digi-Key. Спочатку Beagle board розроблявся у тісній співпраці з спільнотою open source з метою демонстрації можливостей системи на мікропроцесорі OMAP3530. Та зараз підтримка даного пристрою не велика тому розробка для нього займає більше часу.

Raspberry Pi – одноплатний комп'ютер, розроблений британським фондом Raspberry Pi Foundation. Хоча його початкове призначення — стимулювати вивчення базових комп'ютерних наук у школах, з ростом популярності та можливостей платформи він став застосовуватись і в реальних проектах. На raspberry може бути встановлений на вибір один з декількох дистрибутивів Linux. Зазвичай використовується RaspbianOS.

Orange pi - це одноплатний комп'ютер із відкритим вихідним кодом. Він може працювати з Android 4.4, Ubuntu, Debian, Raspbian Image. Хоча розробники заявляють, про широкі можливості та підтримку спільноти, на ділі виявляється що багато простих речей розробникам доводиться допилювати самим[14]. Зокрема є проблеми з драйверами від виробника, які працюють не найкращим чином. Багато розробників також жалуються на відсутність повної документації, що змушує їх використовувати

техніки реверс-інжинірінгу для того аби дізнатись як же працює та чи інша функціональність.

Arduino – апаратна обчислювальна платформа для аматорського конструювання, основними компонентами якої є плата мікроконтролера з елементами вводу/виводу та середовище розробки на мові програмування, що є спрощеною підмножиною C/C++. Arduino може використовуватися як для створення автономних інтерактивних об'єктів, так і підключатися до програмного забезпечення, яке виконується на комп'ютері (наприклад: Processing, Adobe Flash, Max/MSP, Pure Data, SuperCollider). Інформація про плату (схема друкованої плати, специфікації елементів, програмне забезпечення) знаходяться у відкритому доступі і можуть бути використані тими, хто воліє створювати плати власноруч[15].

Stm32 – це плати із мікроконтролером, побудованим на ядрі ARM Cortex-M3 [16]. Дане ядро має багато переваг, які будуть перераховані нижче, але його основна перевага на сьогоднішній день - універсальність. За два роки Cortex-M3 став індустріальним стандартом. Про це свідчить те, скільки виробників, що приєдналися до даної архітектури. Всі основні виробники мікроконтролерів, мають або розвивають рішення на основі цієї архітектури: STMicroelectronics, Texas Instrument, NXP, ATMEL, Analog Devices, Renesas і т.д. Але компанія ST одна з перших випустила свої мікроконтролери Cortex-M3 і швидко стала домінуючим гравцем на цьому ринку. Це говорить про високу якість і привабливості рішень STMicroelectronics. Помітне і значне зростання продажів самих ядер Cortex-M3. Аде є певні обмеження даної платформи: це повільний порівняно з іншими час розробки, складність налаштування.

Однією з вимог до системи годування є захищеність комунікації між сервером та пристроєм. Оскільки комунікація здійснюватиметься через мережу інтернет, то найбільш безпечним способом є використання протоколу https. https потребує певної кількості обчислювальних ресурсів, що не кожна з перерахованих вище систем має. І зважаючи на можливості перерахованих мінікомп'ютерів та плат, raspberry pi є найкращим кандидатом для системи годування тварин.

Зважаючи на наступні особливості Raspberry Pi було обрано як основа для системи, що проектується:

- відносна дешевизна;
- універсальність по напрямкам застосування;
- велика підтримка спільноти, що дає можливість пришвидшити розробку;
- підтримка декількох мов програмування;
- можливість використання <https>.

4.1.2 Використання сервоприводів

У пристрої системи годування контейнер з кормом повинен мати дозатор та певну механічний засув. Засув повинен закриватись та відкриватись на деякий час. Дозатор повинен видавати певну мінімальну кількість корму. Для таких механічних маніпуляцій підходять сервоприводи.

Сервопривод, або сервомотор - механічний привід з автоматичною корекцією стану через внутрішню негативний зворотний зв'язок, відповідно до параметрів, заданими ззовні.

Сервоприводом є будь-який тип механічного приводу (пристрою, робочого органу), що має в складі датчик (положення, швидкості, зусилля тощо) і блок управління приводом (електронну схему або механічну систему тяг), автоматично підтримує необхідні параметри на датчику відповідно до заданого ззовні значення. (положенню ручки управління або чисельним значенням від інших систем)[16].

Простіше кажучи, сервопривід є «автоматичним точним виконавцем» - отримуючи на вхід значення керуючого параметра (в режимі реального часу), він ґрунтуючись на показаннях датчика прагне встановити і підтримувати це значення на виході виконавчого елемента.

До сервоприводів, як до категорії приводів, відноситься безліч різних регуляторів і підсилювачів з негативним зворотним зв'язком, наприклад, гідро-, електро-, пневмопідсилювачі ручного приводу керуючих елементів (зокрема, рульове управління і гальмівна система на тракторах і автомобілях), проте термін «сервопривід» найчастіше використовується для позначення електричного приводу зі зворотним зв'язком по положенню, що застосовується в автоматичних системах для

приводу керуючих елементів і робочих органів.

Сервоприводи в даний час застосовуються в високопродуктивному обладнанні наступних галузей: машинобудування; автоматичні лінії виробництва: напоїв, упаковки, будматеріалів, електроніки і т. д., підйомно-транспортна техніка; поліграфія; деревообробка, харчова промисловість [17].

Так як датчик у складі сервоприводу зазвичай контролює елемент, що рухається, електричний сервопривід має наступні переваги перед кроковим двигуном:

- не пред'являє особливих вимог до електродвигуна і редуктора - вони можуть бути практично будь-якого потрібного типу і потужності (а крокові двигуни, як правило, малопотужні і тихохідні);
- гарантує максимальну точність, автоматично компенсуючи: механічні (люфти в приводі) або електронні збої приводу;
- поступовий знос приводу, шаговому ж двигуну для цього потрібно періодична калібрування;
- теплове розширення приводу (при роботі або сезонне), це було однією з причин переходу на сервопривід для позиціонування головок в жорстких дисках;
- забезпечуючи негайне виявлення відмови (виходу з ладу) приводу (з механічної частини або електроніці);
- велика можлива швидкість переміщення елемента (у крокової двигуна найменша максимальна швидкість в порівнянні з іншими типами електродвигунів);
- витрати енергії пропорційні опору елемента (на кроковий двигун постійно подається номінальна напруга з запасом по можливим перевантаженням).

Існують синхронні та асинхронні сервоприводи. Синхронні здатні виставити швидкість обертання електродвигуна з великою точністю, так само як прискорення і кут повороту. Синхронні види приводів можуть швидко досягати номінальної швидкості обертання. Асинхронні здатні точно витримувати швидкість навіть на дуже низьких оборотах.

Сервоприводи принципово розділяють на електромеханічні і електрогідромеханічні. Електромеханічні приводи складаються з редуктора і

електродвигуна. Але їх швидкодія виявляється набагато менше. У електрогідромеханічних приводах рух створюється шляхом руху поршня в циліндрі, внаслідок чого швидкодія виявляється на дуже високому рівні. [18]

Враховуючи властивості крокових двигунів та сервоприводів, можна сказати що для цілей системи годування підійдуть електромеханічні синхронні сервоприводи так як вони фіксують певний заданий кут повороту ротора, і тому що мають хороший обертовий момент.

4.1.3 Вибір модулю керування сервоприводами

Для системи годування важливо мати 2 сервоприводи. Один з яких контролюватиме та відміряти правильну кількість корму, і інший закриватиме та відкриватиме засув для висипання корму.

Важливість другого сервоприводу для засуву велика, так як саме вона дозволить забезпечити пристрій від зазіхань тварини на корм. Саме вона має створювати бар'єр запаху.

Raspberry pi має обмежені можливості по керуванню сервоприводами. Та щоб не городити велосипеди варто використати модуль PCA9685 I2C[19], що дасть змогу під'єднати на одну лінію I2C декілька сервоприводів.

Даний модуль є платою розширення на базі 12-розрядного ШІМ PCA9685, до якої можна підключити до 16 сервоприводів. Регулюється частота ШІМ в діапазоні від 24 до 1526 Гц. Також з використанням додаткового модуля з'являється можливість послідовно підключити до 62 плат або 992 сервоприводів. За допомогою даної плати з'являється можливість розробляти як прості, так і складні проекти [19].

Модуль спроектований на основі мікросхеми PCA9685 в корпусі TSSOP28. Плата модуля має 6 висновків, де VCC використовується для подачі живлення на мікросхему плати, GND - «земля», а на контакт V + подається напруга від джерела живлення. Висновок SCL - лінія тактування, а SDA - лінія даних. Для управління виходами використовується контакт OE. За допомогою даного контакту можна відключати висновки, переводячи їх в один з логічних рівнів.

Плата розширення управляється за допомогою I2C. Інтерфейс I2C підключається через контакти SCL і SDA. У разі, якщо необхідно використовувати більш ніж 16 сервоприводів, на платі розташовані додаткові контакти GND, OE, SCL, SDA, VCC, V + до яких можна підключити додатковий модуль. Після підключення додаткового модуля необхідно присвоїти йому унікальну адресу, так як базові адреси плат є 0x4. Адреса плати вказується на виходах A0, A1, A2, A3, A4 і A5. Присвоїти унікальну адресу можна за допомогою адресних перемичок, які знаходяться в правому верхньому куті плати.

Підключаються сервоприводи по 3 контактам (V +, PWM і GND). Два виходу V + і GND забезпечують харчування (до 6 В), а висновок PWM зраджує ШІМ сигнали.

Живлення плати і виходів ШІМ розділене і може бути 3 - 5 В. ШІМ харчування можна подавати як на висновки V +, так і на затискачі за допомогою клем зовнішнього джерела живлення. На платі встановлений фільтруючий конденсатор, який усуває перешкоди при роботі модуля з великими навантаженнями.

Для цього модуля вже існують драйвера та програмні пакети для управління ним, що надалі спростить з ним роботу.

4.1.4 Вибір операційної системи для пристрою

Ubuntu Core – це ужата, суворо обмежена та повністю транзакційна операційна система. Розробники розробляли її з самого початку зосереджуючи увагу на безпеці та спрощеному обслуговуванні для приладів та великих мереж пристроїв. Ubuntu Core працює за допомогою оснащення - універсального формату упаковки Linux. Може бути встановлена на Raspberry[20]. Перевагами даної системи є захищеність, верифікація всіх компонентів, довга підтримка протягом мінімум 10 років. Недоліками ж є – не повна підтримка всіх пакетів, необхідність власноруч реалізовувати деякі прості речі.

Fedora Internet of Things - це дистрибутив Fedora, орієнтований на те, щоб бути сильним фундаментом для екосистем IoT. Незалежно від того, чи працюєте ви над проектом вдома, промисловими шлюзами, розумними містами або аналітикою з AI /

ML, Fedora IoT забезпечує надійну платформу відкритого коду, на якій можна розвиватись. Fedora IoT випускається регулярно щоб допомогти вам тримати свою свою екосистему в актуальному стані[21]. Fedora підтримує пристрої Raspberry Pi Model 2B та 3-серії пристроїв, включаючи 3B, 3A у Fedora 29 та пізніших випусках.

Хоча базові можливості raspberry вже підтримуються, Fedora IoT має невелику спільноту, що створює для неї компоненти.

Raspbian - заснована на Debian операційна система для Raspberry Pi. Існує кілька версій Raspbian, в тому числі Raspbian Stretch і Raspbian Jessie. З 2015 року Raspbian офіційно представлена Raspberry Pi Foundation в якості основної операційної системи для одноплатних комп'ютерів Raspberry Pi. Raspbian була створена Майком Томпсоном і Пітером Гріном в якості незалежного проекту. Перша збірка була виконана у 2012. Операційна система знаходиться в стадії активної розробки. Raspbian оптимізована для низькопродуктивних процесорів ARM, які використовуються в лінійці комп'ютерів Raspberry Pi[23].

Raspbian має вбудовану підтримку мови програмування Python, менеджера пакетів pip та велику підтримку спільноти розробників відкритого програмного коду.

4.1.5 Вибір мови програмування для пристрою

Підтримка мови програмування python в Raspbian OS дає переваги використання мови високого рівня, при цьому маючи змогу управляти виконавчими механізмами та периферією пристрою.

Python - високорівнева мова програмування загального призначення, орієнтована на підвищення продуктивності розробника і читання коду. Синтаксис ядра Python мінімалістичний. У той же час стандартні бібліотеки включають великий обсяг корисних функцій [23].

Python підтримує структурне, об'єктно-орієнтоване, функціональне, імперативне і аспектно-орієнтоване програмування. Основні архітектурні риси - динамічна типізація, автоматичне керування пам'яттю, повна інтроспекція, механізм обробки виключень, підтримка багатопоточних обчислень, високорівневі структури даних.

Підтримується розбиття програм на модулі, які, в свою чергу, можуть об'єднуватися в пакети.

Python активно розвивається як мова програмування, нові версії з додаванням чи зміною мовних властивостей виходять приблизно раз в два з половиною роки. Вона не піддавалася офіційній стандартизації, роль стандарту де-факто виконує CPython, що розробляється під контролем автора мови, але вже існують підвиди такі як Numba. На даний момент Python займає третє місце в рейтингу ТЮВЕ з показником 8,5%. Аналітики відзначають, що це найвищий бал Python за весь час його присутності в рейтингу[23].

Pip - це інсталятор пакетів для Python. Ви можете використовувати pip для встановлення пакетів з індексу пакетів Python та інших індексів. Вважається інсталятором пакетів за замовчуванням.

Для підсистеми пристрою системи годування тварин, знадобиться декілька пакетів:

- Crontab;
- Requests;
- Adafruit_servokit.

Crontab – пакет, що має методи розбору виразів cron та визначення наступної дати і часу та перевірки чи відповідає поточний момент виразу cron. Зокрема, він обчислює затримку в секундах від моменту, коли викликається метод next, до того, як елемент слід виконати далі [24].

Даний пакет дозволить розпарсити записи cron, та запустити годування за графіком цих записів. За задумкою, графік годування складатиметься з декількох записів cron.

Requests - це елегантна та проста HTTP-бібліотека для Python, створена для людей. Requests дозволяє надсилати HTTP запити надзвичайно легко. Не потрібно вручну додавати рядки запитів до своїх URL-адрес або формувати-кодувати ваші POST-дані. Увімкнено збереження та об'єднання HTTP-з'єднань автоматично, завдяки urllib3 [25].

Adafruit_servokit – бібліотека для роботи з модулем PCA9685 I2C[19].

Представляє розробнику інтерфейс, що спрощує комунікацію та робить всі налаштування для керування модулем всередині себе.

4.2 Обґрунтування вибору компонентів серверної частини

4.2.1 Вибір фреймворку для серверної частини

Серверна частина системи потребує фреймворк, що дозволить забезпечити всі необхідні вимоги системи. Різні сучасні web-фреймворки сьогодні пропонують схожу функціональність і загалом вибір залежить лише від мови програмування [26]. У даній роботі розглядається створення серверної частини з допомогою мови програмування C#.

ASP.NET Core - це крос-платформна, високопродуктивна система з відкритим кодом для створення сучасних додатків, підключених до Інтернету на основі хмарних технологій. За допомогою ASP.NET Core можливо створювати веб-додатки та служби, програми IoT та мобільні програми. Використовувати засоби розробки для Windows, macOS та Linux. Розгорнути в хмару або локально. Запуск у .NET Core або .NET Framework.

Чому варто обрати ASP.NET Core? ASP.NET Core - це редизайн ASP.NET 4.x, з архітектурними змінами, що призводять до більш гнучкого, більш модульного фреймворку, порівняно із попередниками.

ASP.NET Core надає такі переваги [26]:

- уніфікований підхід для створення веб-інтерфейсу та веб-API.
- архітектурно створений для полегшення тестування;
- можливість розробки та роботи на Windows, macOS та Linux;
- інтеграція сучасних фреймворків клієнтської частини та сучасних технологій розробки;
- орієнтованість на розгортання у хмарі;
- легкість, високопродуктивність та модульність системи обробки HTTP-запитів;
- можливість розміщення на: Kestrel, IIS, HTTP.sys, Nginx, Apache, Docker;
- інструменти, що спрощують сучасну веб-розробку.

4.2.2 Вибір Nuget пакетів

Важливим інструментом будь-якої сучасної платформи розробки є механізм, за допомогою якого розробники можуть створювати, ділитися та використовувати корисний код. Часто такий код поєднується в "пакети", які містять складений код разом з іншим вмістом, необхідним в проектах[27].

Для .NET (включаючи .NET Core) механізмом спільного використання коду спільного використання Microsoft є NuGet, який визначає, як пакети для .NET створюються, розміщуються та використовуються, а також надає інструменти для кожної з цих ролей.

Простіше кажучи, пакет NuGet - це єдиний ZIP-файл, який містить скомпільований код (DLL), інші файли, пов'язані з цим кодом, та описовий маніфест, що включає інформацію, наприклад номер версії пакета. Розробники з кодом для спільного використання створюють пакети та публікують їх на публічному або приватному хості. Споживачі пакетів отримують ці пакунки від відповідних хостів, додають їх до своїх проектів, а потім викликають функціональність пакета у коді проекту.

4.2.2.1 Telegram bot framework

Telegram bot framework - це простий фреймворк для побудови ботів Telegram. Ідеально підходить для запуску декількох ботів всередині однієї програми ASP.NET Core. Цей пакет націлений на .NET Standard 1.6 та .net Core.

Створення бота з хорошою архітектурою стає дуже простим за допомогою цього фреймворку.

Особливості фреймворку:

- дозволяє мати декілька ботів, що працюють в одному додатку;
- можливість ділитися кодом (оновлення обробників) між декількома ботами;
- проста у використанні веб-хуків (спеціально з розгортанням Docker);
- оптимізовано для створення Telegram-ігор;

- спрощує багато повторюваних завдань у розробці ботів.

Даний пакет використовує інший пакет Telegram bot, що - найпоширеніший .NET-клієнт для Telegram Bot API.

4.2.2.2 NCrontab.Advanced

На стороні серверу також потрібно обробляти вирази cron, наприклад, щоб показати заплановані на сьогодні годування, тому доцільно використати nuget-пакет NCrontab.Advanced.

NCrontab.Advanced - це бібліотека, написана на C # для .NET Standard Library 1.0, яка забезпечує такі можливості:

- Розбір виразів cron
- Форматування виразів cron
- Розрахунок подій часу на основі графіка cron

Ця бібліотека надає можливості аналізу, форматування та алгоритм для створення подій на основі розкладу дат, вираженого у форматі cron, наприклад '0 12 * */2 Mon'.

FluentScheduler – власне планувальник та виконавець задач для asp.net core. Автоматизований планувальник роботи з fluent інтерфейсом. Він потрібний для запуску на задньому плані задач.

4.2.2.4 Entity Framework

Entity Framework Core (EF) - це легка, розширювана, відкрита та кросплатформна версія популярної технології доступу до даних Entity Framework.

EF Core слугує маппером об'єктів з реляційних баз даних (ORM), дозволяючи розробникам .NET працювати з базою даних за допомогою .NET-об'єктів і виключати необхідність писати багато коду доступу до даних, який їм зазвичай потрібно писати. EF Core підтримує безліч різних баз даних. Тобто можна приєднатись з допомогою нього як до MSSQLServer, Oracle Db, Postgres, Sqlite чи іншої реляційної БД.

5 БУДОВА СИСТЕМИ

В даній роботі розглядається розробка двох підсистем системи годування, а саме: пристрою та серверної частини. Для серверної частини розглядається інтеграція із зовнішньою системою Telegram.

5.1 Структурна схема підсистеми пристрою

Пристрій в даній системі виконує операції: видача корму, відправка статистики на сервер, отримання графіків та команд із серверу, надсилання попереджень. Для повноцінного функціонування пристрою, необхідна наявність бездротової мережі (IEEE802.11).

Підсистема пристрій складається із наступних компонентів:

- Raspberry Pi;
- PCA9685 I2C модуль[19];
- сервоприводи: для дозатору корму, для засуву;
- динамік;
- цифрова камера;
- датчик дна контейнеру;
- кнопки: ручного годування, налаштування;
- вимикач.

Всі компоненти відображені на структурній схемі пристрою (додаток X). А в наступних підрозділах їх буде описано детально.

5.1.1 Raspberry Pi

Raspberry Pi – «мозок» підсистеми пристрій. Його задачею є керування всіма іншими компонентами та обмін даними із сервером.

Для отримання інформації із елементів периферії Raspberry Pi має 40 контактів,

що дозволяє під'єднати багато виконавчих механізмів пристроїв різного призначення.

Деякі групи контактів мають спеціальне призначення, наприклад можуть разом працювати по принципу певного протоколу обміну даними. Наприклад, в нашому випадку, є необхідність в інтерфейсі I2C для зв'язку із модулем PCA9685. Для цього використовуються лише 4 контакти, але на самому модулі ще шістнадцять трійок контактів, що дозволяють підключити до шістнадцяти сервоприводів. В даній системі буде використовуватись лише два.

Окрім відкритих контактів загального призначення, на платі Raspberry пі розташовані також гнізда спеціального призначення: 3.5мм minijack, USB. Вони будуть використовуватись для компонентів системи. Камеру під'єдніється до порту USB. А для взаємодії використовується пакет `opencv`[29].

На платі запускається операційна система Raspbian. Це дає змогу використовувати пакети та створювати код на різниц мовах програмування включаючи `c++` та `python`.

5.1.2 Модуль PCA9685 та сервоприводи

Даний модуль підключається до системи з допомогою чотирьох контактів і отримує дані по інтерфейсу I2C. До нього підключаються два сервоприводи.

Перший сервопривод призначений для дозування корму. У режимі очікування корм знаходиться у контейнері, до якого приєднаний сервопривід. Сервопривід зсуваючи дно контейнеру, наповнює мірну чашу (певного мінімального розміру), згодом зсуває назад і висипає корм у наступний відсік.

Другий сервопривод слугує механізмом відкриття-закриття засуву, що повинен щільно закриватись. Його призначення приховувати запах корму, щоб домашня тварина не намагалася вгризтись в чи пошкодити середину пристрою.

З мінікомп'ютера модуль PCA9685, а отже сервоприводи керуються за допомогою пакету `adafruit_servokit`[30].

5.1.3 Кнопки та датчик закінчення корму

Для кнопок та датчика закінчення корму будуть використовуватись порти загального призначення Raspberry Pi. Для забезпечення стабільної роботи та уникнення помилок при налаштування, кожна кнопка буде під'єднуватись до порту з допомогою 2 резисторів як показано на рисунку 5.1.

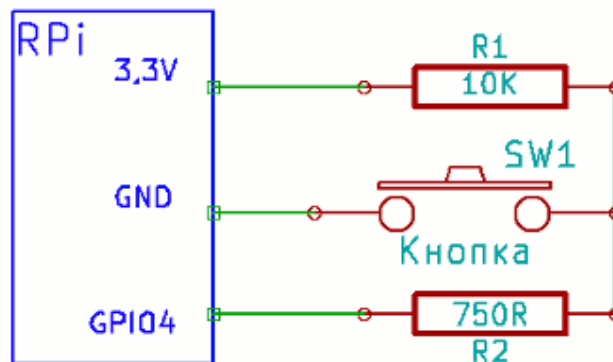


Рисунок 5.1 – Схема під'єднання кнопки[31]

Таке підключення є надлишковим, але в разі прямого підключення є ймовірність підпалити використовуваний вихід, оскільки при ініціалізації системи (завантаження мінікомп'ютера) або невірної установці режимів порту (налаштований на вихід) на використовуваний пін може піти напруга +3.3 В, що в свою чергу при замиканні кнопки викличе коротке замикання порту з землею GND[31].

Для обробки натискання кнопки та натискань датчика закінчення корму використовується бібліотека RPi.GPIO. Кнопки налаштовуються на вхід. Наявність напруги на вході - вважається, що кнопка неактивна, відсутність – кнопка нажата.

5.2 Діаграма компонентів серверної частини

Основа серверної частини системи – це веб сервер, реалізований на технології ASP.NET Core [26]. Саме він здійснює обмін даними між Telegram ботом та пристроєм користувача.

Для зберігання даних сервер використовує реляційне сховище даних MSSQL. У наступних розділах буде розроблено таблиці, схему бази даних для потреб системи. Для зберігання тимчасових даних, наприклад дані сесії, команди, що йдуть від користувача пристрою, та втрачають актуальність, якщо вчасно не доставлені, використовується серверний кеш.

Сервер взаємодіє із зовнішнім світом, тобто Telegram ботом, чи пристроєм через API. Для бота – серверна частина залежить від Telegram Bot API[11], а для пристрою реалізований REST API. Пристрій повинен запитувати оновлення сервера, та застосовувати їх до себе. Спрощено діаграма компонентів наведена на рисунку 5.2.

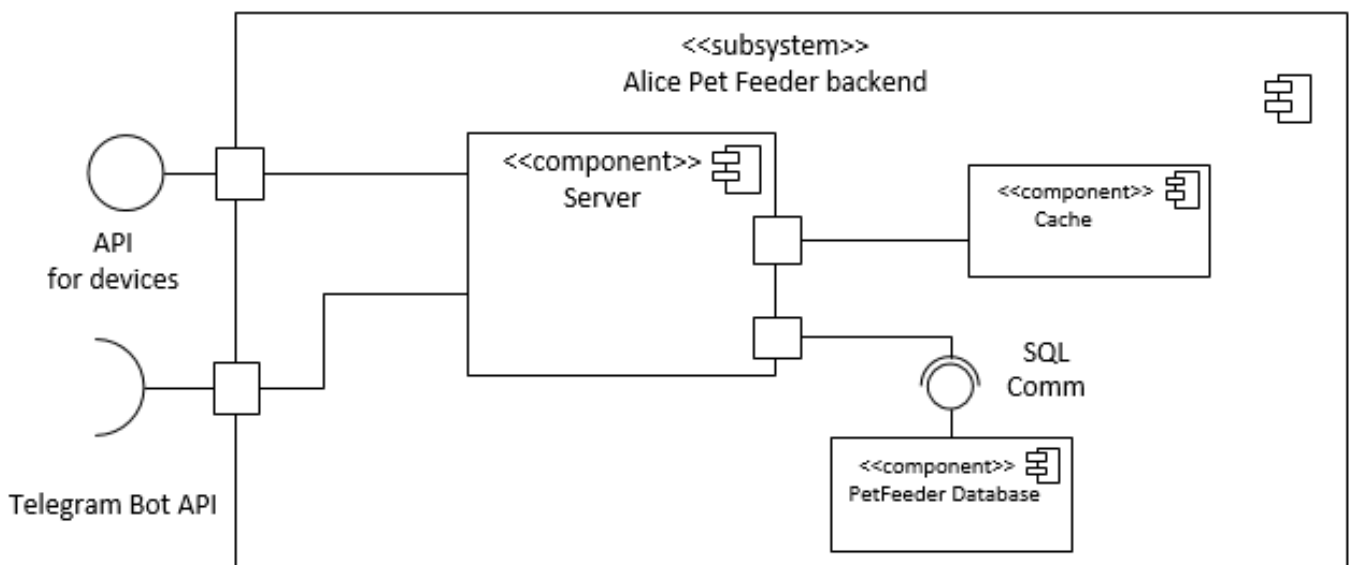


Рисунок 5.2 – Спрощена діаграма компонентів

Компоненту Server складається із п'ятих головних dll файлів: Alice.PetFeeder.Web.dll, Alice.PetFeeder.Application.dll, Alice.PetFeeder.Business.dll, Alice.PetFeeder.Persistence.dll, Alice.PetFeeder.TelegramBot.dll. Головною є Web.dll (тут і далі наводитимуться назви бібліотек dll без приставки Alice.Petfeeder), обробка всіх комунікацій починається саме в ній. Вона в свою чергу залежить від TelegramBot.dll, і код, що відповідає за комунікації щодо обміну даними з ботом знаходиться саме в ній. Web.dll має залежність від всіх інших. Application.dll – оголошує інтерфейси, що використовуватимуться по всій системі, зокрема інтерфейси роботи зі сховищем. Persistence.dll – містить логіку інтеграції із

конкретним сховищем даних, його ініціалізацію, перевірку тощо. Зокрема реалізує інтерфейси рівня доступу до даних. `Business.dll` – містить найголовнішу логіку серверної частини по маніпуляції із даними доменної області.

Всі описані вище зв'язки відображені на детальній діаграмі компонентів серверної частини (Додаток X).

6 РОЗРОБКА СХЕМИ БАЗИ ДАНИХ

6.1 Опис предметної області

Система годування тварин передбачає автоматизацію процесу годування, а отже створення графіків із встановленим часом подачі їжі. Графіків може бути декілька, але активним лише один. Важливо забезпечити можливість зміни графіку (виключень із правил стандартного годування) та відключення графіку зовсім. Зокрема можливість перемикання між різними графіками. Кожен графік може мати тип як от: звичайний, дієта тощо. Користувач повинен мати змогу встановити власну назву. Також графік може мати встановлений за замовчуванням розмір порції. Можливе існування графіків, що не прив'язані до конкретного пристрою – приклади, що користувач може скопіювати для свого пристрою (дієти до прикладу). Маючи на увазі, пристрій можливо ще не завантажив створений чи відредагований графік, можливо варто зберігати дані про те, який графік наразі діє на пристрої.

В межах графіку годування може бути встановлений різний час годування для різних днів, для цього використовуватиметься вираз `stop`[32]. Кожний встановлений час годування може мати відмінний від встановленого в графіку розмір порції. Графік на сьогодні може вираховуватись із декількох `stop` виразів, тому можливо варто забезпечити їх типи наприклад: щоденний, обраний день тижня, конкретна дата та довільний вираз – для універсальності.

При виготовленні пристрою, дані про нього повинні бути занесені до бази даних. Для валідації пристрою, при синхронізації даних. Користувач має мати змогу керувати налаштуваннями декількох пристроїв. Зокрема, налаштовувати графіки годувань, встановити назву для кожного із свої пристроїв, дозволити іншим користувачам доступ до налаштувань конкретного пристрою. Передбачити можливість того, що у компанії з'являться нові моделі пристроїв.

Кожне годування по графіку чи ручне повинно бути збережено у журнал годувань. Таким чином користувач матиме змогу перевірити скільки разів сьогодні вже годували тварину, та побачити статистику.

Система має мати змогу під'єднання користувачів з різних зовнішніх систем,

наприклад різних месенджерів. Користувач має мати змогу обирати зручну для себе інтеграцію із зовнішньою системою. Спеціальні для зовнішньої системи дані користувача, що потрібні для інтеграції повинні зберігатися.

6.2 Аналіз предметної області

Виходячи з потреб предметної області, можна виділити наступні сутності:

- Графік годування (Schedule);
- Рядок часу (TimeRow);
- Користувач (User);
- Пристрій (Device);
- Журнал годування (FeedLog);
- Зовнішня система (ExternalSystem).

Інфологічна модель бази даних (таблиці та їх атрибути) системи годування тварин матиме вигляд:

- Графік годування (Ідентифікатор, Назва, Активовано, Розмір порції за замовчуванням, Максимальна кількість їжі на день, Режим дієти);
- Рядок часу (Ідентифікатор, Ранг, Вираз стон, Розмір порції);
- Користувач (Ідентифікатор, Ім'я, Прізвище, Номер телефону, Імена в зовнішніх системах);
- Пристрій (Ідентифікатор, Тип пристрою, Штрих-код, Дата виробництва, Назва, Час останнього з'єднання з сервером);
- Журнал годувань (Ідентифікатор, Дата+час, Розмір порції);
- Зовнішня система (Ідентифікатор, Назва, Активність).

Зв'язки між сутностями матимуть наступний вигляд:

- Пристрій (0..N) використовується (0..N) Користувачем;
- Користувач (0..N) має доступ до (0..N) Пристрою;
- Графік (0..N) застосовується до (0..1) Пристрою;
- Пристрій (0..1) має (0..N) Графік;

- Графік (1) складається з (0..N) Рядок часу;
- Рядок часу (0..N) входить в (1) Графік;
- Пристрій (1) має (0..N) Журнал годувань;
- Журнал годувань (0..N) відправлений з (1) Пристрою;
- Зовнішня система (0..N) дозволяє взаємодію (0..N) Користувачу.

Інфологічна модель бази даних зображена на ER-діаграмі (див. Додаток Г).

6.3 Схема БД

Інфологічна модель є основою подальшого проектування БД. Деякі сутності та їх зв'язки потребують декомпозиції та нормалізації.

Оскільки сутності Користувач та Пристрій мають зв'язок типу «багато-до-багатьох», то необхідно ввести додаткову таблицю Доступ користувача (UserDeviceAccess), таким чином розбивши одне відношення на два з типом «один-до-багатьох». Зокрема нова таблиця міститиме маркер того, чи є користувачем адміністратором пристрою.

Аналогічним чином відношення між сутностями Користувач та Зовнішня система повинне бути розбите на два з типу «один-до-багатьох» та введена нова таблиці ЗовнішняСистема-Користувач (ExternalSystemUser). Атрибутами даного відношення є дані про користувача, що необхідні для зовнішніх систем. Наприклад, Ідентифікатор користувача в зовнішній системі (UserIdOnExternalSystem). Сутність ЗовнішняСистема-Користувач матиме такі атрибути: Ідентифікатор користувача, ідентифікатор зовнішньої системи, Ідентифікатор користувача в зовнішній системі, Увімкненість інтеграції для користувача, Пріоритет.

З результатами аналізу предметної області, декомпозиції та нормалізації було створено схему БД (Додаток Х). Опис таблиць, полів та їх призначень наведений нижче (Таблиці 6.1 – 6.9).

Таблиця 6.1 – DeviceType

Назва поля	Тип даних	Обмеження	Призначення
DeviceTypeId	int	PK	Ідентифікатор типу пристрою
Name	nvarchar(100)	not null	Назва моделі

Таблиця 6.2 – Device

Назва поля	Тип даних	Обмеження	Призначення
DeviceId	int	PK	Ідентифікатор пристрою
SerialNumber	nvarchar(100)	unique, not null	Штрихкод
DeviceTypeId	int	FK	Посилання на тип пристрою (модель), екземпляром якого є даний пристрій
ManufacturedDate	date	not null	Дата виробництва
Name	nvarchar	not null	Назва пристрою дана користувачем
LastOnlineUtcDttm	datetimeoffset	not null	Дата останнього зв'язку з пристроєм

Таблиця 6.3 – Schedule

Назва поля	Тип даних	Обмеження	Призначення
ScheduleId	int	PK	Ідентифікатор графіку
DeviceId	int	FK, null	Посилання на пристрій, до якого застосовується графік
Name	nvarchar(100)	null	Назва дана користувачем
IsEnabled	bit	not null	Чи увімкнений графік?

IsLoadedToDevice	bit	not null	Чи надіслано до пристрою
DefaultPortionSize	smallint	not null	Розмір порції за замовчуванням, застосовується якщо не встановлено окремо для кожного рядку часу
MaxFoodSizePerDay	int	not null	Максимальна кількість їжі на день, застосовується у строгому режимі дієти
DietMode	smallint	not null	Режим дієти
LastUpdateUtcDttm	datetimeoffset	not null	Дата останнього редагування
LastUpdateUserId	int	not null	Користувач, що востаннє редагував

Таблиця 6.4 – TimeRow

Назва поля	Тип даних	Обмеження	Призначення
TimeRowId	int	PK	Ідентифікатор рядка часу (годування)
ScheduleId	int	not null	Посилання на пристрій, до якого застосовується графік
Rank	smallint	not null	Група (0 - Щоденно, 10 - День тижня, 20 - Конкретна дата, 30 - Будь-який вираз cron). Коли застосовується графік, для визначення годування буде застосовано всі рядки часу

			найвищої групи, що має сьогодні хоча б одне заплановане годування
CronExpression	nvarchar(1000)	not null	Вираз cron. Наприклад "0 0 6 * * ? *" - що значить "щодня о 6й ранку"
PortionSize	smallint		Розмір порції, кратний мінімальній кількості, яку може видати пристрій

Таблиця 6.5 – FeedLog

Назва поля	Тип даних	Обмеження	Призначення
FeedLogId	bigint	PK	Ідентифікатор запису журналу годування
DeviceId	int	not null	Посилання на пристрій, до якого застосовується запис журналу
DttmUtc	datetimeoffset	not null	Дата годування
Portion	smallint	not null	Розмір порції, кратний мінімальній кількості, яку може видати пристрій

Таблиця 6.6 – ExternalSystem

Назва поля	Тип даних	Обмеження	Призначення
ExternalSystemId	int	PK	Ідентифікатор зовнішньої

			системи
Name	nvarchar(200)	not null	Назва зовнішньої системи
IsActiveIntegration	bit	not null	Чи увімкнена інтеграція із системою

Таблиця 6.7 – User

Назва поля	Тип даних	Обмеження	Призначення
UserId	int	PK	Ідентифікатор користувача
FirstName	nvarchar(100)	not null	Назва зовнішньої системи
LastName	nvarchar(100)	not null	Чи увімкнена інтеграція із системою
Phone	nvarchar(50)		Телефон користувача

Таблиця 6.8 – ExternalSystemUser

Назва поля	Тип даних	Обмеження	Призначення
ExternalSystemUserId	int	PK	Ідентифікатор запису
ExternalSystemId	int	FK, not null	Посилання на зовнішню систему
UserId	int	FK, not null	Посилання на запис користувача
UserIdInExternalSystem	nvarchar(100)	not null	Нікнейм/логін за яким можна ідентифікувати користувача у зовнішній системі
TelegramChatId	bigint		Ідентифікатор чату з нашим ботом в Telegram
Enabled	bit	not null	Чи увімкнено інтеграцію
Priority	smallint	not null	Пріоритет, на випадок неможливості застосувати

			даний канал комунікації
--	--	--	-------------------------

Таблиця 6.9 – UserDeviceAccess

Назва поля	Тип даних	Обмеження	Призначення
UserDeviceAccessId	int	PK	Ідентифікатор запису
UserId	int	FK, not null	Посилання на запис Користувача
DeviceId	int	FK, not null	Посилання на Пристрій, до якого відноситься даний запис про доступ
IsAdminUser	bit	not null	Маркер користувача, що є адміністратором пристрою. Користувач, що першим зв'язав свій аккаунт з пристроєм - стає адміністратором

Додатково варто зазначити, що деякі таблиці потребують створення індексів, так як очікується велика кількість даних в них. Таблиця FeedLog (Журнал годувань) буде опитуватись для статистики годувань певного пристрою, тому планується створити індекс для двох її полів DeviceId, DttmUtc. Що дасть змогу швидше отримувати дані.

7 РЕАЛІЗАЦІЯ БІЗНЕС-ЛОГІКИ СИСТЕМИ

7.1 Багатошарова архітектура

Більшість традиційних програм .NET розгортаються як одиничні виконуваний файл, або як одна веб-програма, що працює в одному додатку IIS чи іншого веб-сервера. Це найпростіша модель розгортання і вона дуже добре слугує для багатьох внутрішніх систем та систем публічного доступу. Однак, навіть з огляду на те, що це єдина одиниця розгортання, більшість нетривіальних бізнес-додатків отримують перевагу від логічного поділу на кілька шарів[33].

З точки зору організації коду нескладних додатків, простіше всього розмістити весь код в одному модулі, але в цьому разі, зі збільшенням розміру та складності проекту кількість файлів та папок також зростатиме. Тоді бізнес-логіка стане розсіяна між папками, і не буде чітких ознак, які паки (враховуючи їх як структурні одиниці) від яких папок мають залежати, а від яких ні. Ця відсутність організації на рівні проекту часто призводить до спагеті коду[34]. Для вирішення цих проблем програми часто перетворюються на багатомодульні рішення, де кожен модуль знаходиться у певному шарі програми.

Оскільки додатки зростають у складності, одним із способів управління цією складністю є розбиття програми відповідно до обов'язків модулів чи проблем. Це слідує принципу розділення відповідальності[35] і може допомогти підтримувати зростаючу базу коду, щоб розробники могли легко знайти, де реалізована певна функціональність чи схована певна логіка.

Багатошарова архітектура пропонує ряд переваг, крім простої організації коду. Впорядковуючи код у шари, спільні методи чи іншу функціональність низького рівня можна повторно використовувати у вищих шарах та у всій програмі. Це повторне використання корисне, оскільки означає, що потрібно писати менше коду і тому, що це дозволяє модулю концентруватись на реалізації, дотримуючись принципу "не повторювати себе"[36].

Завдяки багатошаровій архітектурі у додатку можуть застосовуватись обмеження щодо того, які шари можуть спілкуватися з іншими шарами. Це допомагає

досягти інкапсуляції. Коли шар змінюється або необхідно замінити його реалізацію, це повинно впливати лише ті шари, з якими він пов'язаний залежністю. Обмежуючи, які саме шари залежать від даного, вплив змін можна пом'якшити, щоб одна зміна не вплинула на всю роботу всього додатку[33].

Шари програми значно полегшують заміну функціональності в програмі. Наприклад, додаток може спочатку використовувати базу даних SQL Server для збереження чи навіть зберігання у пам'яті, але пізніше може вибрати стратегію збереження на основі хмари або одну з нереляційних технологій. Якщо програма належним чином інкапсулювала свою стійкість у логічному рівні, цей рівень написаний з використання реалізації SQL може бути замінений новим, що реалізує той же публічний інтерфейс.

Окрім потенціалу заміни реалізацій у відповідь на майбутні зміни вимог, додаткові шари також можуть спростити підміну реалізації з метою тестування. Замість того, щоб писати тести, які працюють з реальним рівнем(шаром) даних або з рівнем користувальницького інтерфейсу програми, ці шари можна підмінити під час виконання тестів фальшивими реалізаціями [37], які надають відомі відповіді на запити. Це, як правило, робить тести набагато простішими для написання та набагато швидшими для запуску в порівнянні з запущеними тестами проти реальної інфраструктури програми.

З огляду на описані вище переваги багатошарової архітектури, вона була взята за основу реалізації серверної частини системи годування. За підходом «чистої архітектури»[38] було відділено в окремі модулі та реалізовано наступні шари :

- шар доменної області (Business layer);
- шар логіки додатку (Application layer);
- шар доступу до даних (Persistence layer);
- шар взаємодії із користувачем (Presentation layer), що представлений проектом TelegramBot (Рис.7.х);
- шар взаємодії із пристроєм, що представлений проектом Web.

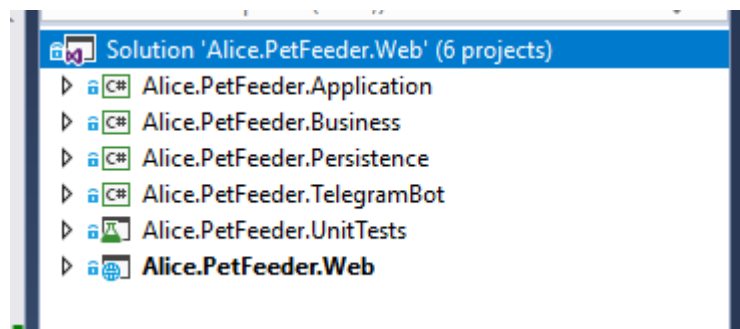


Рисунок 7.1 – Структура рішення

Виділені шари співвідносяться за принципом «цибулевої архітектури»[39].
 Рис.7.2. Чим ближче шар до ядра тим із вищим рівнем абстракції він працює. Чим далі шар від ядра ти більш конкретні взаємодії та реалізації він описує. Кожен більший шар залежить від меншого – попереднього.

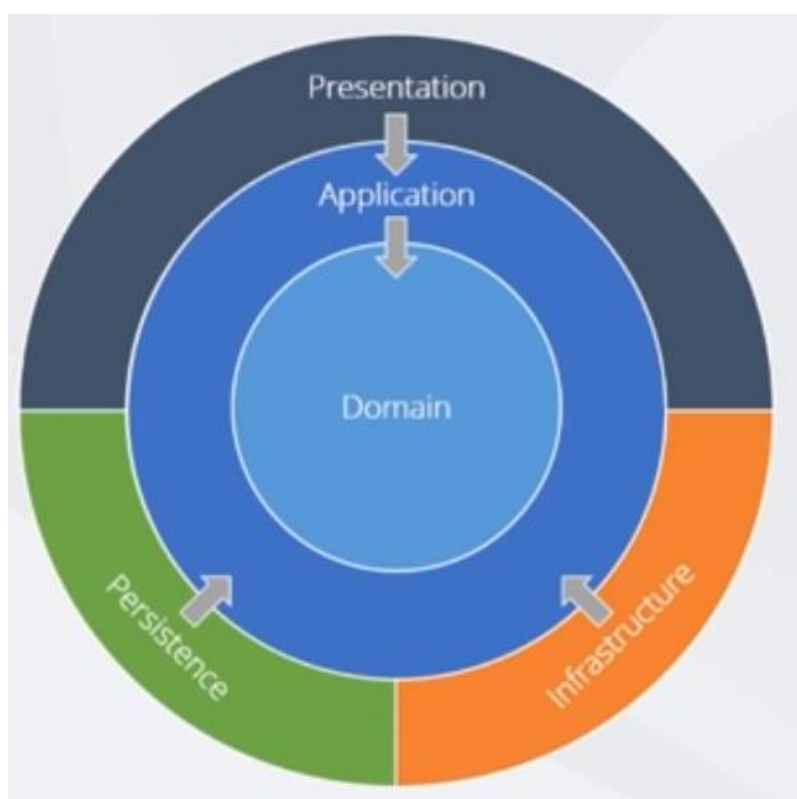


Рисунок 7.2 - «Цибулева архітектура»(Onion architecture) [40]

Шар доменної області - зберігає класи об'єктів доменної моделі та сервіси із правилами доменної області. Тут містяться найбільш загальні правила та правила найвищого рівня абстракції. Вони найменш вірогідно можуть змінитися, коли щось

ЗОВНІШНЄ ЗМІНИТЬСЯ.

Класи сутностей, що оголошені в шарі доменної області:

- Device;
- DeviceType;
- ExternalSystem;
- ExternalSystemUser;
- FeedLog;
- Schedule;
- TimeRow;
- User;
- UserDeviceAccess.

Класи сервісів, що оголошені в шарі доменної області:

- ScheduleService;
- TimeRowService.

Шар логіки додатку – є в деякому сенсі ключовою ланкою, тому що саме тут оголошуються інтерфейси для більш зовнішніх рівнів архітектури (шар доступу до даних, користувацького інтерфейсу та інших шарів взаємодії із навколишнім світом). В даному шарі оголошуються багато спільних речей для додатку, що не залежатимуть від реалізації більш зовнішніх шарів: виключення, команди та запити доменної області.

Загальні інтерфейси, що оголошені в шарі логіки додатку:

- ICacheService;
- ICurrentUserService;
- INotificationService;
- IPetFeederDbContext.

Оголошені виключення, що можуть бути оброблені в більш зовнішніх шарах:

- BadRequestException;
- DeleteFailureException;
- NotFoundException;
- ValidationException.

Шар доступу до даних – описує реалізацію раніше оголошеного інтерфейсу IPetFeederDbContext. Та містить декілька допоміжних класів, що будуть використані при запуску цього серверного додатку та зв'яжуть реалізацію із оголошеним раніше інтерфейсом. Містить класи:

- DependencyInjection;
- ModelBuilderExtensions;
- PetFeederDbContext.

Шар взаємодії із користувачем - в даному рішенні це проект TelegramBot. Він описує взаємодію веб-серверу із сервісом Telegram. Використання nuget пакету Telegram Bot Framework дозволило суттєво зменшити об'єм роботи по інтеграції з даною платформою. Тут реалізовані інтерфейси, що оголошені в даному пакеті та необхідні для роботи Telegram бота:

- CustomBotOptions;
- FeederBot;
- Startup_FeederBot;
- BotServiceProvider.

Також проект TelegramBot оголошує обробники команд, що приходять. Кожна з команд реалізовує базовий клас BaseCommand із nuget пакету Telegram Bot Framework:

- TodaysScheduleCommand;
- FeedNowCommand;
- ScheduleCommand;
- StartCommand ;
- ButtonsCommand;
- HelpCommand.

А для кнопок, що телерам дозволяє показати користувачеві, оголошені та реалізовані обробники, що реалізують інтерфейс IUpdateHandler того ж пакету:

- ChooseDeviceHandler;
- DeleteScheduleTimeRowHandler;
- DeleteTodayTimeRowHandler;

- DisableScheduleHandler;
- EnableScheduleHandler;
- FeedDeviceHandler.

7.2 Паттерни, що застосовуються

Для реалізації принципів проектування програмних систем SOLID[35]. Це сприятиме створенню такої системи, яку буде легко підтримувати і розширювати протягом довгого часу [35]. Принципи SOLID - це настанови, які також можуть застосовуватися під час роботи над існуючим програмним забезпеченням для його поліпшення - наприклад для видалення «погано пахнучого коду». Тому в серверній частині системи годування використовуються відповідні паттерни та підходи.

7.2.1 Inversion of control

Принцип Inversion of control (IoC) полягає у написанні слабо зв'язаного коду. На практиці це реалізується так що кожен компонент системи був ізольованим від інших і не покладався у своїй роботі на деталі конкретної реалізації інших компонентів[41]. Одною з практик, що реалізують цей принцип є «ін'єкція залежностей»(Dependency Injection). В ASP.NET Core це можна використати з допомогою вбудованих можливостей - IoC-контейнеру. Цей контейнер відповідає за зіставлення залежностей з конкретними типами і за впровадження залежностей в різні об'єкти. Тобто коли необхідно отримати об'єкт класу що реалізує певний інтерфейс, то контейнер за інтерфейсом, як параметром, створює його.

Всі налаштування та «зв'язування» абстракцій з реалізаціями проводяться в методі ConfigureServices класу Startup.cs проекту Web. Наприклад у рядку 34 на рис. 7.3. реєструється реалізація інтерфейсу ICurrentUserService.

```

29 public void ConfigureServices(IServiceCollection services)
30 {
31     services.AddPersistence(Configuration);
32     services.AddApplication();
33
34     services.AddScoped<ICurrentUserService, CurrentUserService>();

```

Рисунок 7.3 - Реєстрація реалізації ICurrentUserService

7.2.3 Command query responsibility segregation

Шар логіки додатку оголошує інтерфейси сховищ даних і самі класи для роботи зі сховищами даних. Передбачається що запити до даних суттєво будуть перевищувати кількість команд на модифікацію даних, так як пристрої будуть періодично опитувати сервер на наявність оновлень. В таких умовах доречно використовувати Command query responsibility segregation (CQRS).

Додатково отримуються переваги CQRS[42]:

Незалежне масштабування. CQRS дозволяє масштабувати навантаження читання і навантаження модифікації даних незалежно, що може привести до меншої кількості блокувань.

Оптимізовані моделі даних. Сторона читання може використовувати модель, оптимізовану для запитів, тоді як сторона запису використовує схему, оптимізовану для оновлень.

Безпека. Простіше переконатися, що лише ті що треба об'єкти домену виконують запис у дані. Та лише таким чином, що задуманий і дозволений для даної операції.

Поділ відповідальності. Розмежування сторін читання та запису може привести до отримання більш гнучких і гнучких моделей. Більшість складної логіки бізнесу переходить до моделі запису. Модель читання може бути відносно простою.

А надалі, можна досягти спрощення запитів. Зберігаючи матеріалізований вигляд у базі даних, що лише для читання, програма може уникнути складних з'єднань таблиць під час запитів.

У класах-контролерах asp.net ми могли б робити ін'єкції команд чи запитів і викликати їх. Та зручніше використовувати для цього бібліотек MediatR[43]. З ним

ми можемо фактично замінити всі ці ін'єкції команд та запитів та просто ввести інтерфейс IMediator у базовий клас-контроллер (рис 7.4).

```

7  |  public abstract class BaseController : Controller
8  |  |  {
9  |  |  |  private IMediator _mediator;
10 |  |  |
11 |  |  |  protected IMediator Mediator =>
12 |  |  |  |  _mediator ??
13 |  |  |  |  (_mediator = HttpContext.RequestServices.GetService<IMediator>());
14 |  |  |  }
-- |

```

Рисунок 7.4

В свою чергу запит (рис 7.x рядок 12) GetCurrentScheduleQuery («Отримати поточний графік годування») реалізувати інтерфейс IRequest<> бібліотеки MediatR. А клас обробник цього запиту буде реалізувати інтерфейс IRequestHandler<> (рис 7.5 рядок 16). Бібліотека MediatR зв'яже їх, і при виклику об'єкту IMediator із певним запитом/командою, викличе відповідний обробник цього запиту чи команди.

```

12 |  public class GetCurrentScheduleQuery : IRequest<ScheduleDto>
13 |  |  {
14 |  |  |  public int DeviceId { get; set; }
15 |  |  |
16 |  |  |  public class GetScheduleDetailQueryHandler : IRequestHandler<GetCurrentScheduleQuery, ScheduleDto>
17 |  |  |  {

```

Рисунок 7.5

Таким чином вся логіка по реєстрації запитів та їх обробників була делегована бібліотеці MediatR.

7.3 Процес взаємодії підсистем

7.3.1 Процес взаємодії Telegram із сервером

Взаємодія Telegram із сервером відбувається через Telegram бота. Telegram бот - це спеціальні облікові записи, для встановлення яких не потрібен додатковий номер телефону. Користувачі можуть взаємодіяти з ботами двома способами:

- Надсилати повідомлення та команди ботам, відкриваючи з ними чат або додаючи їх до груп. Це корисно для чатів або ботів новин. Але загалом це використовується майже всіма ботами.

- Надсилати запити безпосередньо з поля введення, ввівши @ім'я бота та запит. Це дозволяє надсилати певний контент із вбудованих ботів безпосередньо у будь-який чат, групу чи канал.

Повідомлення, команди та запити, що надсилаються користувачами, передаються до програмного забезпечення, що працює на серверах Telegram. Їх сервер-посередник обробляє все шифрування та комунікації із API Telegram для. Сервер системи годування повинен спілкуватись із сервером Telegram через простий HTTPS-інтерфейс.

Є два взаємовиключні способи отримання оновлень для бота: метод long polling[4], що може бути викликаний клієнтом з одного боку та webhooks[45] з іншого. Вхідні оновлення зберігаються на сервері, поки бот не отримає їх в будь-який спосіб, але вони не зберігатимуться довше 24 годин [11]. Незалежно від того, який варіант ви обрали, в результаті ви отримаєте серіалізовані в JSON об'єкти.

Метод longpolling getUpdates повинен бути викликаний клієнтом API, за посиланням домену <https://api.telegram.org/{token}>, де token - це реєстраційний номер бота отриманий у BotFather бота. Використовуючи принцип longpolling, встановлюється зв'язок між сервером додатку та сервером Telegram. Цей метод зазвичай застосовують на етапі розробки. А на етапі впровадження, застосовують метод webhooks. Даний метод дещо інший в реалізації та дозволяє зменшити навантаження на мережу, так як 98.5% трафіку в методі longpolling – витрачається дарма[46].

7.3.2 Процес взаємодії серверної частини та пристрою

Процес передачі команд з додатку Telegram відбувається у два кроки. Першим кроком команда з Telegram направляється до серверу. Другим кроком пристрій, який періодично опитує сервер на оновлення, отримує команду.

Для наочності, розглянемо процес передачі команди «годувати зараз». Даний процес показаний на Діаграмі послідовності (Додаток X). Варто зауважити, що на стороні сервера, об'єктами на цій діаграмі виступають клієнт месенджеру Telegram, власне сервер, база даних та кеш.

Спочатку користувач одним із способів надсилає команду з додатку Telegram. Далі перевіряється чи збережено в кеші команду «годувати зараз». Далі йде 2 альтернативні сценарії розвитку подій. Якщо в кеш вже збережене значення, отже заплановано годування. Про це повідомляється користувачу. Зокрема, варто зауважити, що команда у кеші не перебуває вічно.

Далі, якщо ж в кеші немає команди збереженої команди, то посилається запит до бази даних, щоб отримати запис останнього годування, якщо це було нещодавно, користувача запитують підтвердження команди. Підтверджена команда зберігається до кешу.

Після того, як команда збережена на сервері – вона стає готовою для обробки пристроєм. Отже, пристрій, коли запитує оновлення даних, в тому числі графіку годування, отримує і список команд на виконання. Послідовність взаємодії пристрою із сервером зображена на діаграмі послідовності (Додаток X).

Коли пристрій надсилає запит на наявність оновлень, сервер витягує необхідні дані із бази даних, а команди із кешу. В розрізі команди «годувати зараз», якщо в кеші команда відсутня, то у відповіді список команд на виконання пристрою приходить пустим.

Якщо команда присутня в кеші, під час запиту оновлення, то вона надсилається пристрою. Згодом, коли пристрій здійснив годування, він надсилає повідомлення до серверу. Сервер зберігає в базі даних запис, про успішне годування. А значення із кешу видаляється.

8 АЛГОРИТМ РОБОТИ ПРИСТРОЮ

На внутрішній накопичувач пам'яті збережено файли програми. Їх запуск відбувається при старті системи. Для цього використовується стандартний для заснованих на Linux операційних системах планувальник задач cron.

При старті програми, першим кроком відбувається підвантаження всіх бібліотека, від яких вона залежить. Далі відбувається початкове налаштування системи (портів, периферійних пристроїв). Згодом програма починає вічний цикл (Рисунок 7.1).



Рисунок 7.1 – Початок вічного циклу програми, частина команд у циклі

Першим кроком вічного циклу програми є перевірка зв'язку з серверною частиною системи. Якщо зв'язок є, то переходимо до блоку завантаження оновлення даних із серверу. На цьому етапі завантажуються дані графіку годування та список команд, що необхідно виконати.

Наступним кроком є збереження завантажених даних в пристрої, щоб у разі відсутності зв'язку, пристрій міг хоча б за попереднім графіком здійснювати годування.

Далі виконуються команди з сервера. До них належать команди: годувати зараз, зробити знімок. А далі (Рисунок 7.2) викликається процедура «Перевірити графік», що буде розглянута далі.



Рисунок 7.2 – Продовження вічного циклу програми, завершальна частина команд циклу

Якщо система здійснює годування, вона зберігає запис про здійснене годування та за можливості надсилає його на сервер. Згодом інформація про годування стає доступна у вигляді статистики годувань.

Процедура «Перевірити графік» (Рисунок 7.3) має на меті перевірити щойно завантажений, чи раніше збережений графік на заплановані годування. Якщо рядок часу із графіка відповідає саме даному моменту часу, то здійснити годування.

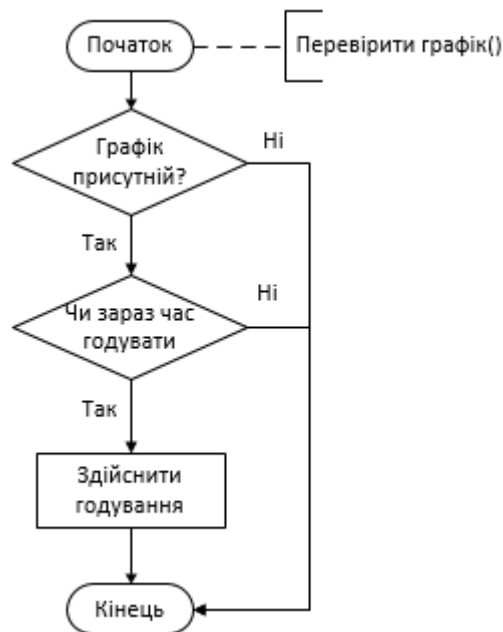


Рисунок 7.3 – Процедура «Перевірити графік»

Процедура «Перевірити графік» починається із перевірки наявності збереженого на пристрої графіка, далі перевіряє даний момент часу на відповідність графіку і здійснює годування. Далі керування повертається коду, який викликав процедуру.

Така функціональність як ручне годування по кнопці, реалізується з допомогою подій у коді програми. Тобто коли кнопка натиснута певний проміжок часу, викликається процес «Здійснити годування». Якщо годування уже здійснюється, то нового не відбудеться.

9 РОЗРОБКА ІНТЕРФЕЙСУ КОРИСТУВАЧА

Для системи годування було вирішено зробити унікальний інтерфейс, використовуючи Telegram. Для цього було зареєстровано бота Telegram. Та створено серверна частина з допомогою ASP.NET Core. Серверна частина отримує всі повідомлення від бота та обробляє їх. В свою чергу апаратна частина, тобто пристрій зв'язується також із серверною частиною.

Відкритий програмний інтерфейс ботів дозволяє взаємодіяти із користувачем як за допомогою повідомлень-команд так і за допомогою зворотній викликів та кнопок. В практичних цілях можна використати різні способи.

9.1 Початок роботи з ботом

Для того, щоб почати роботу з ботом необхідно знайти його в пошуковому відсіку вікна Telegram, ввівши «@alicepetfeederbot». Для того щоб почати роботу з ботом, необхідно ввести команду «start» (Рисунок 9.1).

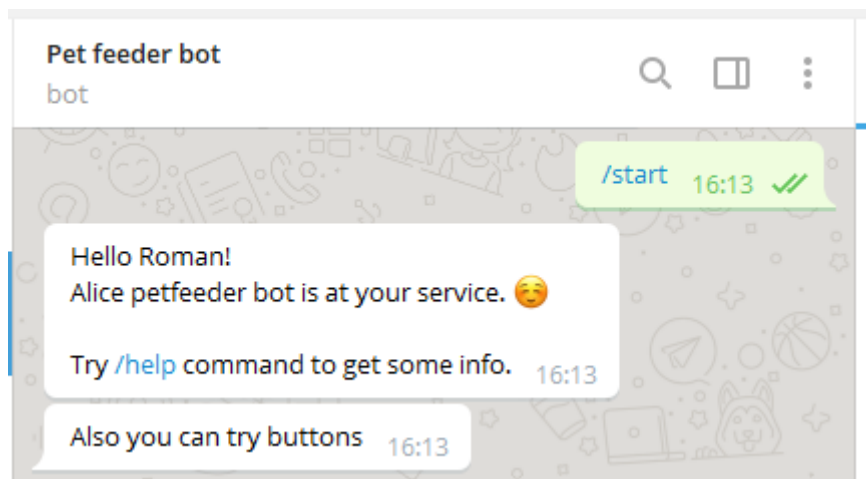


Рисунок 9.1 – Діалог з ботом. Команда «start»

Очікувано, що з самого початку не всі команди доступні, і на серверній частині проводиться валідація введених користувачем команд. Наприклад, на рисунку 9.2, за відсутності у користувача доступу до пристрою, надсилається повідомлення, про те, що

користувач не має жодного пристрою. Отримати доступ користувач може, лише при реєстрації пристрою або якщо інший користувач надав йому доступ до свого.

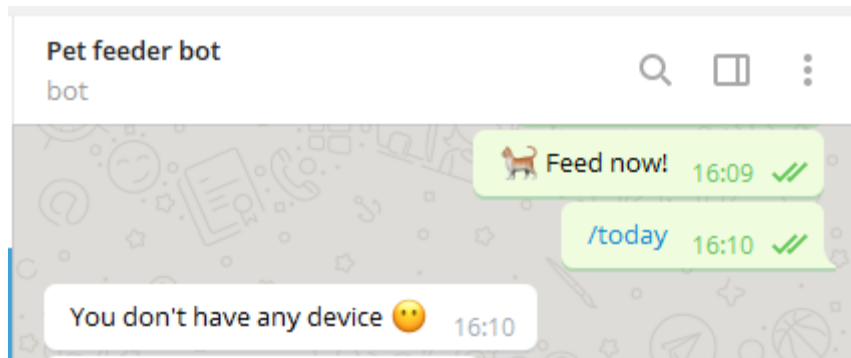


Рисунок 9.2 – Діалог з ботом. За відсутності доступу до пристрою

9.2 Отримання довідки

Для отримання довідкової інформації про бот існує хороша практика створення команд «about» та «help». Команда «about» - надає довідкову інформацію про те, що це за бот, для яких цілей створений та необхідні компоненти для його роботи. На рисунку 9.3 зображено результат виклику даної команди.

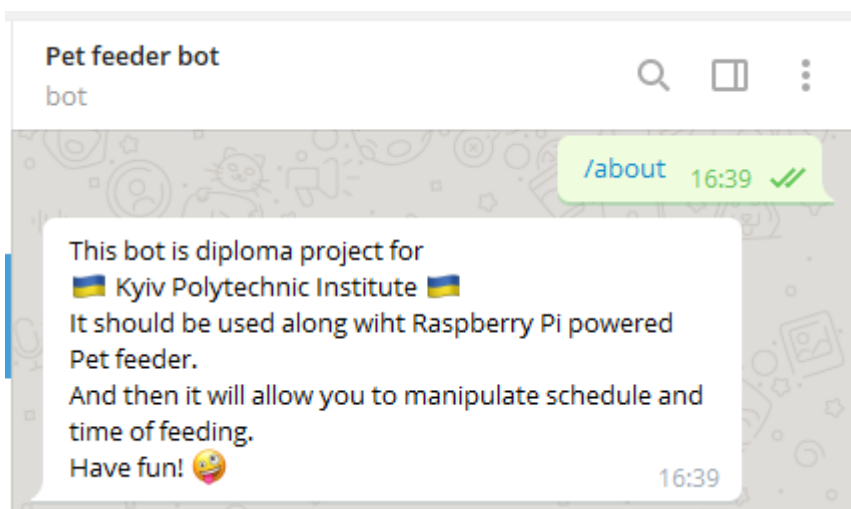


Рисунок 9.3 – Діалог з ботом. Команда «about»

На початку роботи з пристроєм користувачу було запропоновано викликати команду допомоги «help» (Рисунок 9.1). Ця команда дасть змогу користувачеві

ознайомитися із функціоналом бота. У повідомленні-відповіді показується список всіх доступних команд для даного бота. Всі команди підсвічені відмінним від тексту блакитним кольором, щоб користувач міг кліком чи тапом по команді викликати її. Результат виконання команди «help» на рисунку 9.4.

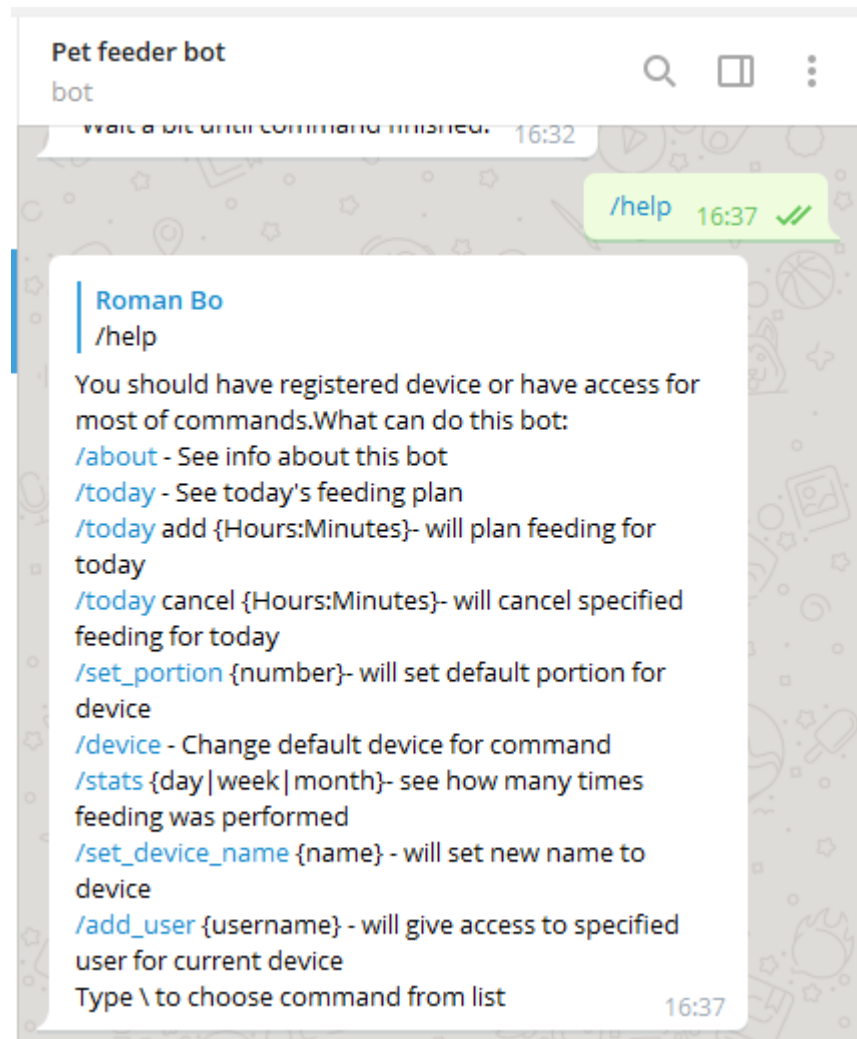


Рисунок 9.4 – Діалог з ботом. Результат команди «help»

9.3 Способи взаємодії з інтерфейсом

Telegram дозволяє розробникам ботів використовувати різні способи взаємодії з користувачами. До них відносяться: текстові повідомлення, команди (текстові повідомлення, що починаються з символу «/»), повідомлення із картинками чи іншими медіа-файлами. Додатково користувач може передати дію боту, якщо бот створить розмітку кнопок під повідомленням. На рисунку 9.5 показано приклад такої

розмітки, що відрізняється від повідомлення. На рисунку 9.5 присутні блоки із часом та блоки із іконкою червоного хреста.

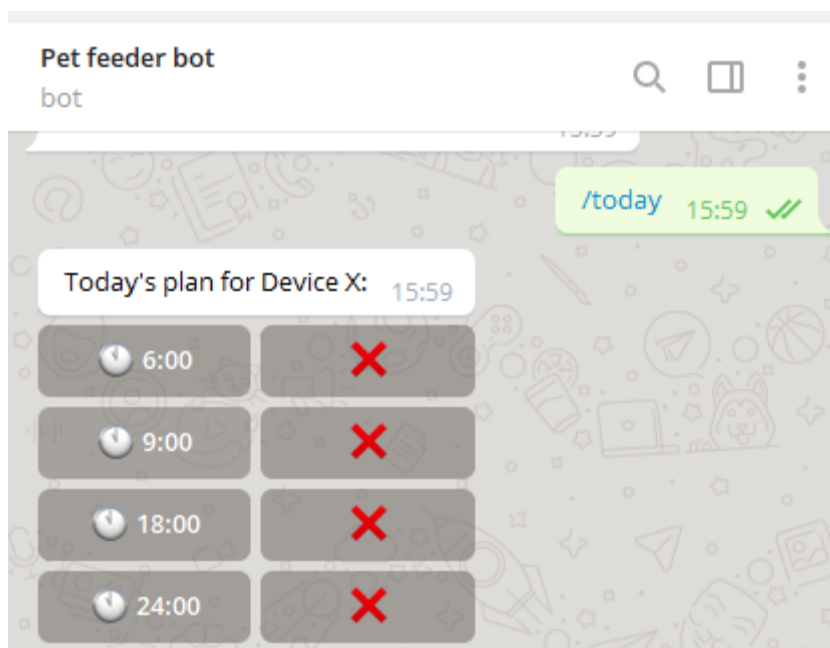


Рисунок 9.5 – Діалог з ботом. Розмітка кнопок у повідомленнях

При натисканні на дані блоки, боту відправляється приховане повідомлення, що у термінології Telegram називається «callback». Користувач не бачить сам текст callback-у, але бачить індикацію у формі годинника, коли він відправляється.

Розробник, при створенні бота може зареєструвати команди для пришвидшення вводу користувачем[47]. Реєстрація команд не впливатиме на їх функціональність, але користувач зможе обрати одну з цих команд із випадального списку. Для активації випадального списку необхідно ввести символ скошеної спраг на ліво риски «/». На рисунку 9.6 продемонстровано дану функціональність для бота, що розробляється у даній роботі, як інтерфейс взаємодії користувача із пристроєм. В основному команди відповідають сценаріям використання системи.

Для команд, як і для всього боту використовується англійська мова, та як з розвитком комп'ютерів вона стала інтернаціональною.

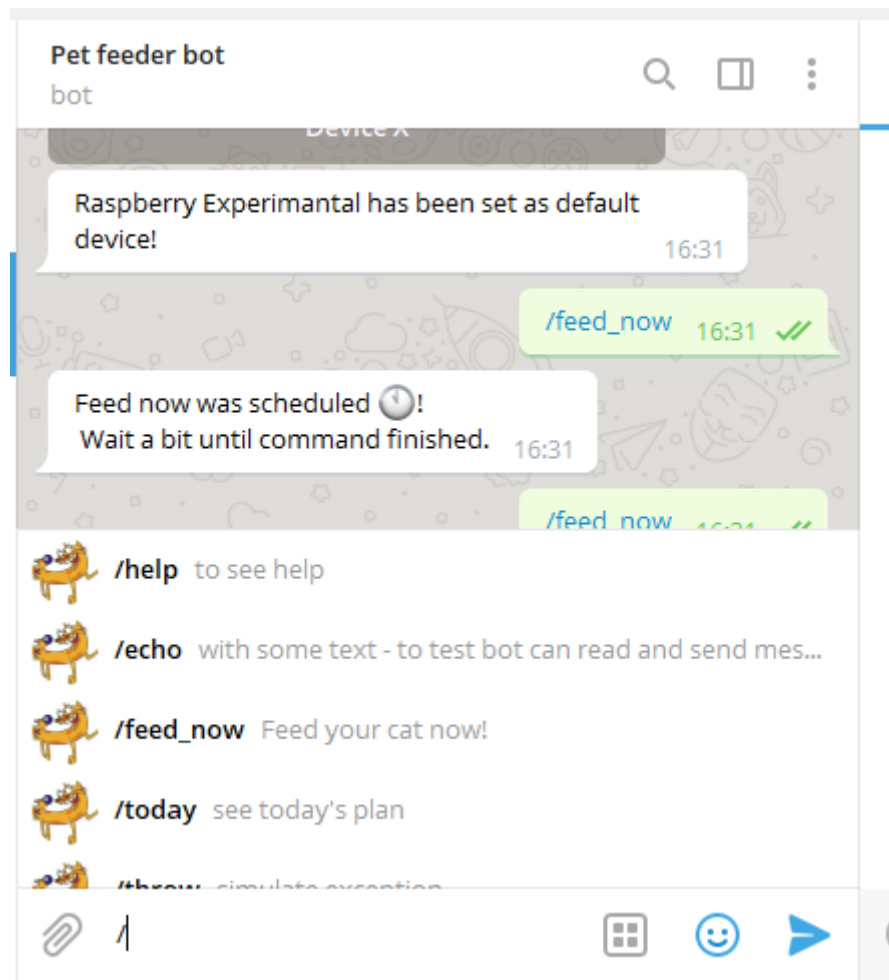


Рисунок 9.6 – Діалог з ботом. Список доступних команд

Також розробнику бота доступна можливість передати у відповіді на повідомлення розмітку кнопок, що стануть доступні користувачу у панелі вводу повідомлення (Рисунок 9.7). На відміну від кнопок під повідомленнями, дані кнопки – це лише шаблони повідомлень. При натисканні на будь-яку з кнопок, користувач відправляє видиме повідомлення із текстом, що зображений на натиснутій кнопці. Зокрема, для боту використовувались символи «емої»[47], так як вони приємні для очей користувача, є більш зрозумілими індикаторами, в деяких випадках, ніж текст, та й досвід користувача, пришвидшує розуміння функціоналу кнопок, якщо на них зображені іконки, а не текст. Кнопки «Feed Now» та «Today's plan» ймовірно будуть використовуватись найчастіше.

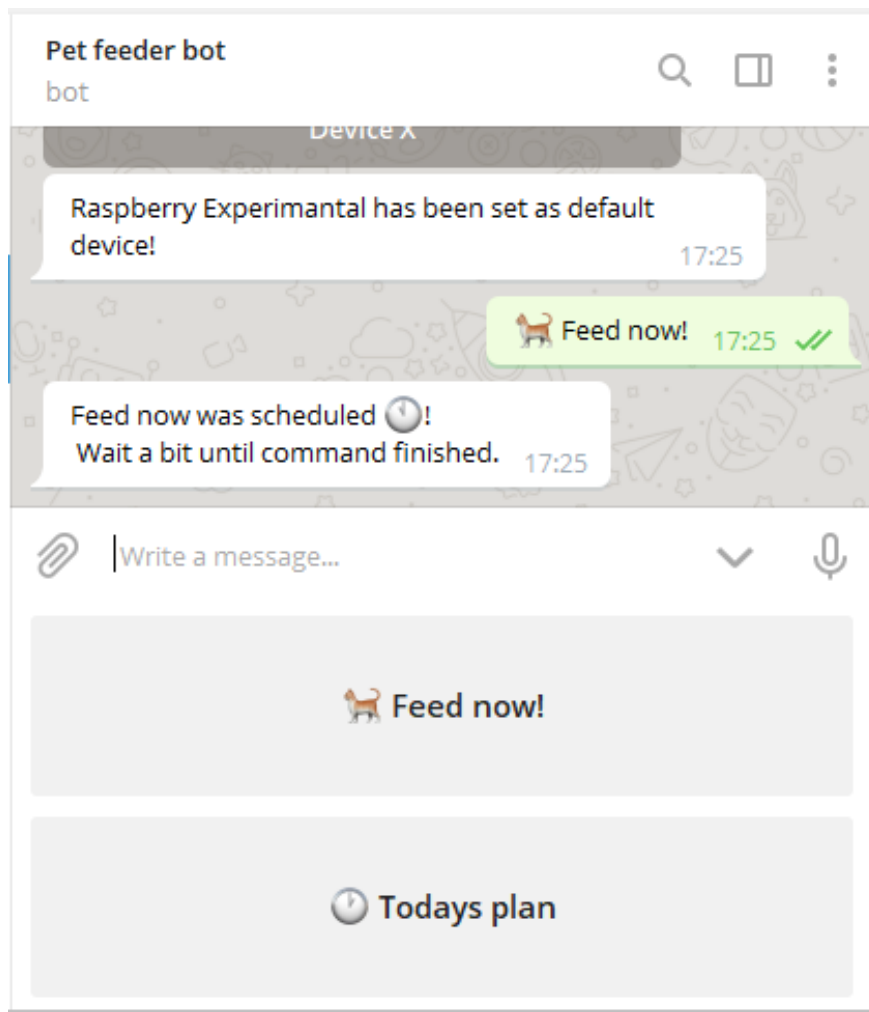


Рисунок 9.7 – Діалог з ботом. Кнопки на панелі вводу повідомлення

9.4 Функціональність команд

Команди є основним способом взаємодії користувача із системою. Список доступних команд можна отримати за допомогою команди «help» (рисунок 9.4) або за допомогою випадаючого списку (Рисунок 9.6).

Для користувача доступні наступні команди:

- device;
- feed_now;
- help;
- start;
- about;
- today;

- today add;
- today cancel;
- set_default_portion;
- stats;
- stats day;
- stats week;
- add_user.

9.4.1 Встановити пристрій за замовчуванням

Команда «device» дає можливість користувачу встановити пристрій за замовчуванням. Таким чином всі подальші команди, що стосуватимуться конкретного пристрою будуть повертати дані та виконувати вказівки користувача саме щодо пристрою за замовчуванням. На рисунку 9.8 показаний процес використання команди «device».

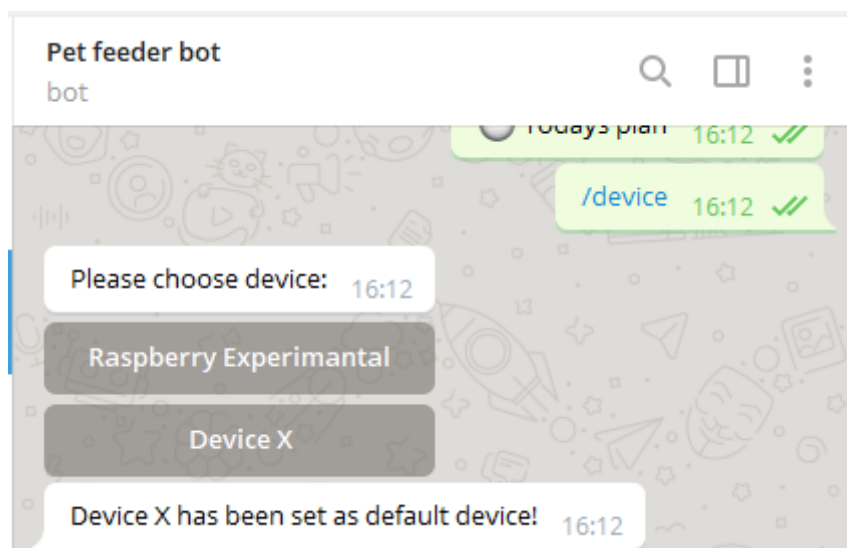


Рисунок 9.8 – Діалог з ботом. Команда «device»

Останнє повідомлення на рисунку 9.8 прийшло у відповідь на натискання кнопки «Device X» під повідомленням «Please choose device:», як підтвердження, що бажаний пристрій обрано як пристрій за замовчуванням.

9.4.2 Графік на сьогодні

Користувачам може бути важливим знати графік годування на сьогодні, особливо, якщо не вони його встановлювали, а було обрано дієту з існуючих. Тому було додано команду «today». На рисунку 9.9 показано роботу команди «today», якщо раніше користувач не обрав пристрій за замовчуванням.

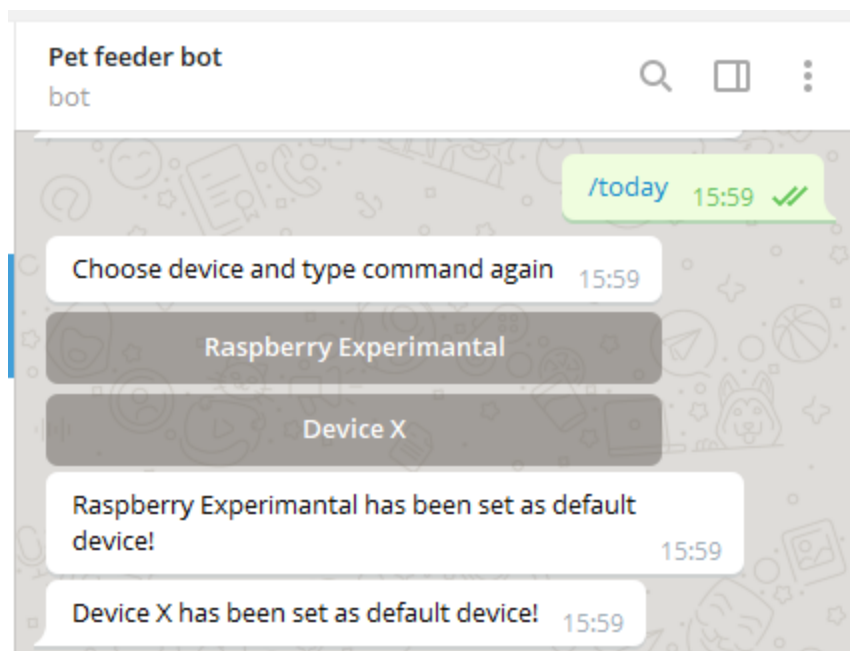


Рисунок 9.9 – Діалог з ботом. Команда «today» просить обрати пристрій за замовчуванням

У випадку, якщо користувач вже обрав пристрій за замовчуванням у результаті команди «device» (Рисунок 9.8), чи в результаті будь-якої команди, яка потребує цього (Рисунок 9.9), то команда «today» покаже графік на сьогодні як на рисунку 9.10. На рисунку видно кнопки із хрестами червоного кольору, натискання на хрести в майбутньому дасть змогу видаляти відповідно поставлені проти них заплановані години годування.

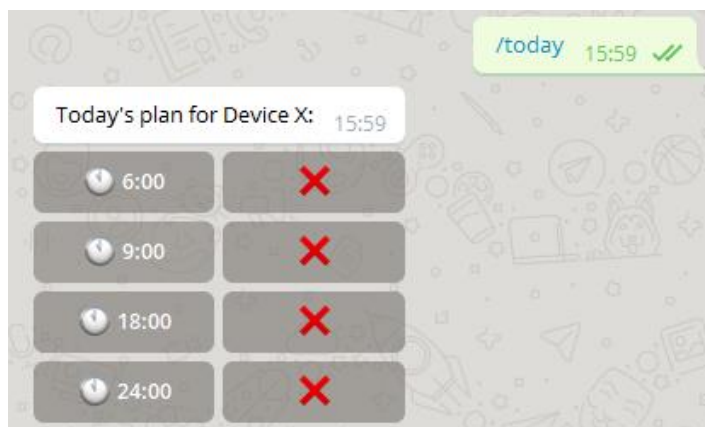


Рисунок 9.10 – Діалог з ботом. Команда «today» показує графік на сьогодні

9.4.3 Модифікації графіку годування

Операції графіку годування дозволяють користувачу налаштовувати графік годування на сьогодні. Для прикладу показано операції додавання та видалення додаткового годування до сьогоднішнього графіку. На рисунку 9.11 зображено результуючий вигляд переписки з ботом, де використані послідовно команди «today add» та команда «today cancel».

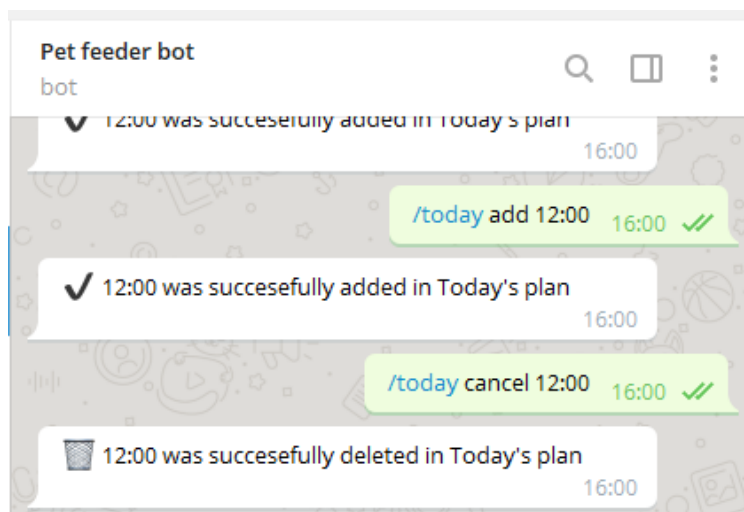


Рисунок 9.11 – Діалог з ботом. Команда «today add» та команда «today cancel»

9.4.4 Команда «Годувати зараз»

Однією з найголовніших команд в системі є команда «годувати зараз» («feed_now»).

Для неї реалізована обробка як командою так і створено кнопку для швидкого виклику даної команди. Кнопку створена, з огляду потенційно високу частоту використання даної команди. На рисунку 9.12 показано процес взаємодії з користувачем під час виклику команди «feed_now».

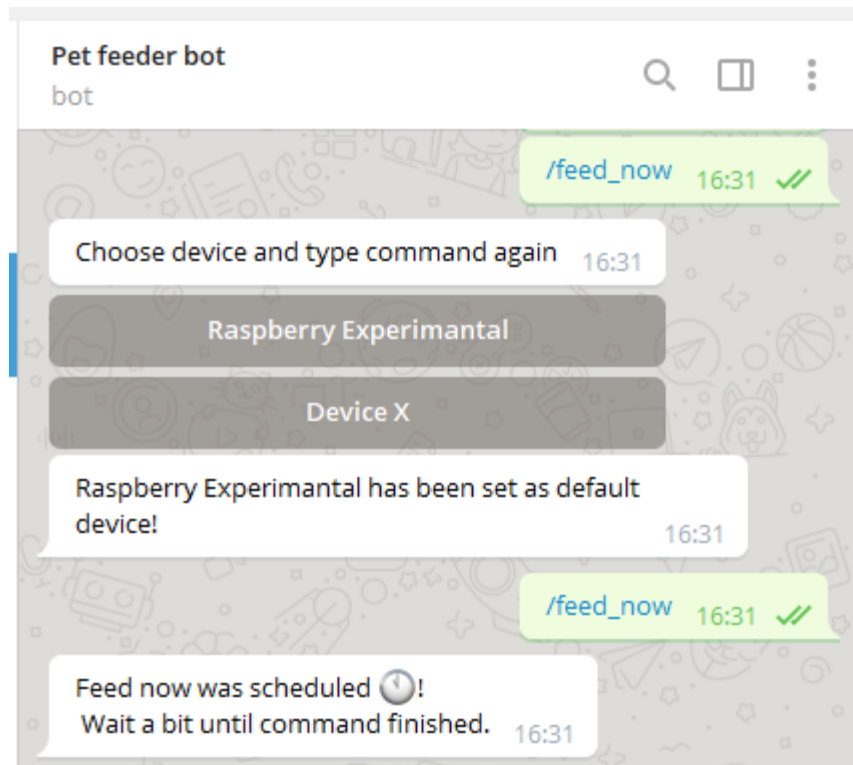


Рисунок 9.12 – Діалог з ботом. Команда «feed_now». Процес взаємодії з користувачем

Якщо користувач протягом короткого часу надішле повторно команду «feed_now», то бот відреагує наступним чином, показаним на рисунку 9.13, він попередить користувача, що нещодавно вже відбулося годування і попросить підтвердити наміри, повторним надсиланням команди.

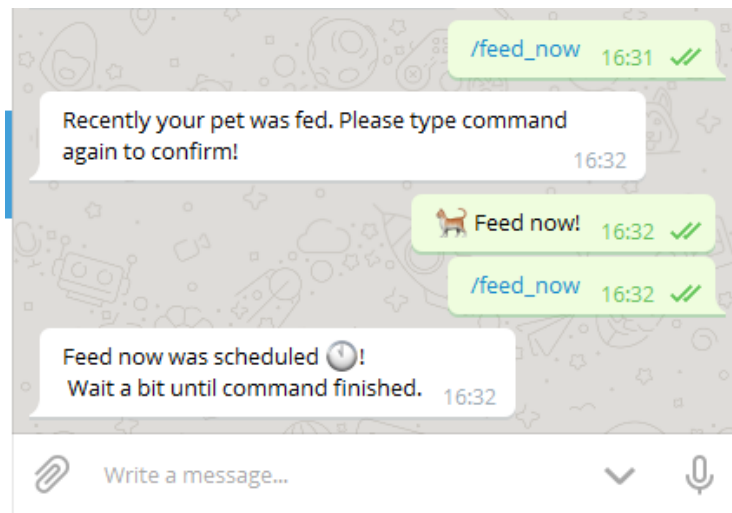


Рисунок 9.13 – Діалог з ботом. Команда «feed_now». Підтвердження годування, якщо пройшло мало часу з нещодавнього годування

9.4.5 Встановлення порції

Розмір порції та можливість його встановлення є одними з ключових можливостей системи. Таким чином досягається здорове харчування домашньої тварини, шляхом уникнення переїдань. На рисунку 9.14 зображено діалог з ботом у випадку виклику команди «set_default_portion».

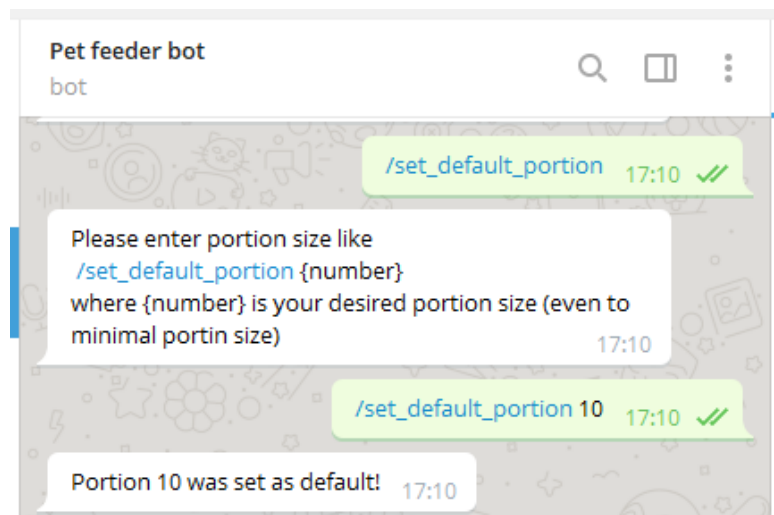


Рисунок 9.14 – Діалог з ботом. Команда «set_default_portion»

9.4.6 Перегляд статистики годувань

За задумом, користувач повинен мати можливість глянути звітність годувань. Це у випадку, якщо не використовувати автоматичний графік годування, а годувати по необхідності це дасть можливість відслідкувати певні тенденції та отримати статистику. Для бота було реалізовано команду «stats», параметром якої є період, за який необхідно витягнути статистику. На рисунку 9.15 зображено діалог з ботом, а саме команда «stats».

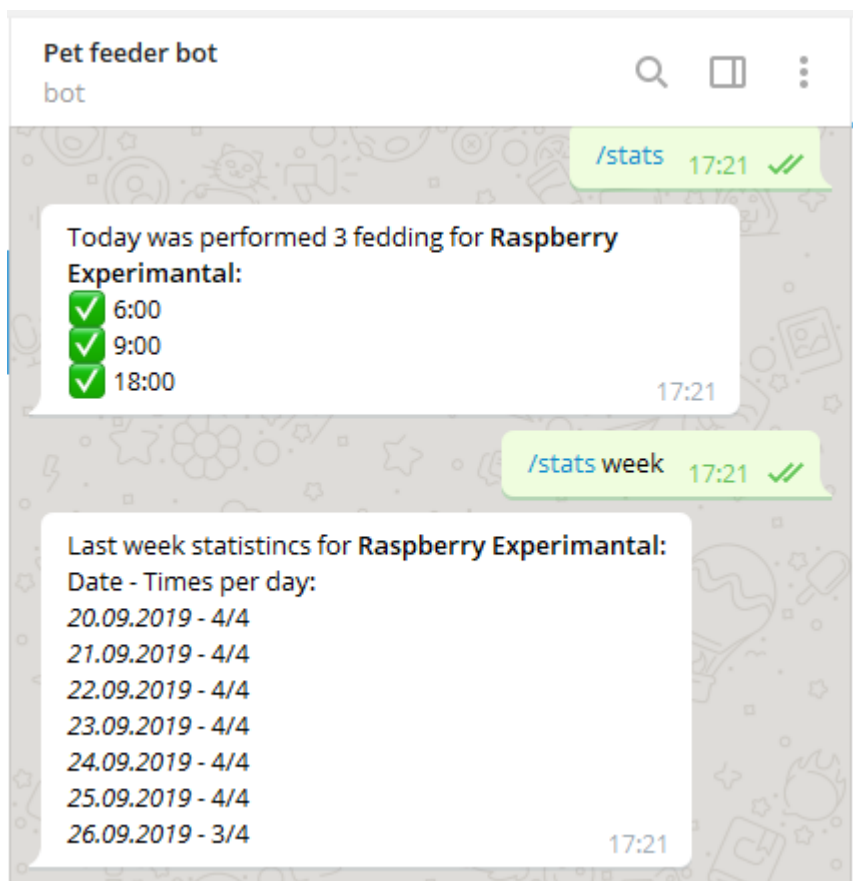


Рисунок 9.15 – Діалог з ботом. Команда «stats»

9.5 Макет пристрою

Задумана система складається із апаратної частини та серверної частини. Апаратна частина, сам пристрій, деякою мірою також має інтерфейс користувача. Елементами цього інтерфейсу можна вважати динамік, індикатор, що реалізується світлодіодом, кнопки. На рисунку 9.16 зображено макет пристрою, де присутні деякі елементи інтерфейсу користувача.

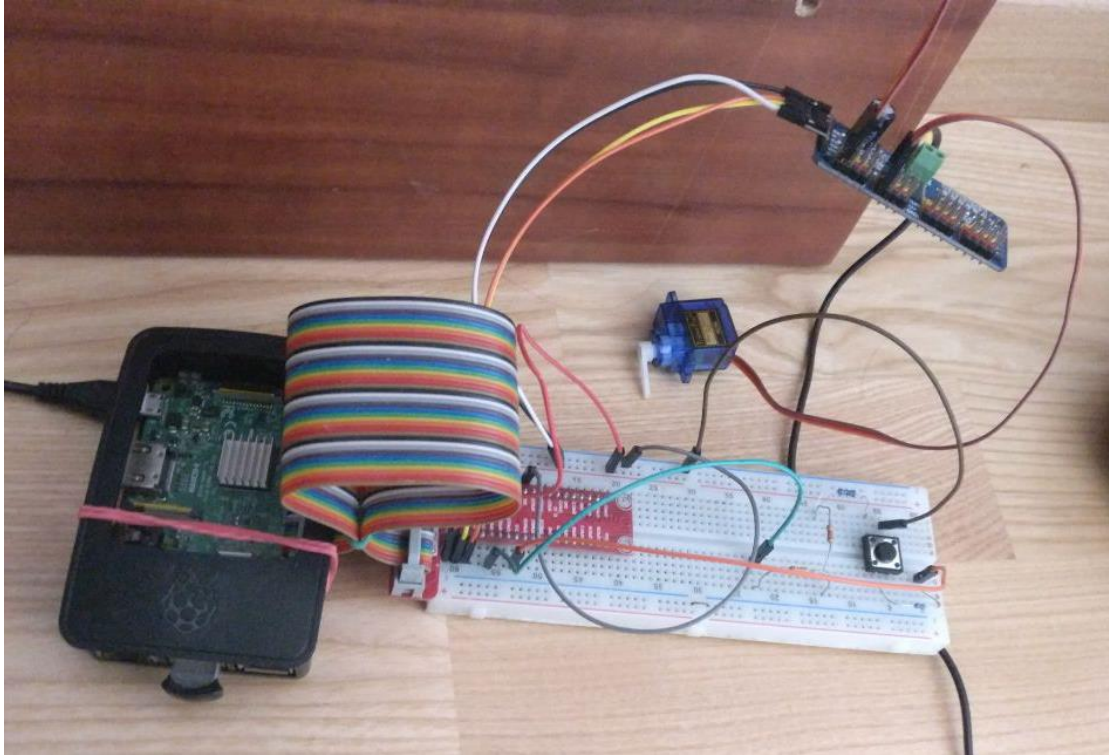


Рисунок 9.16 – Макет пристрою

Кнопки необхідні системі для вмикання, вимикання, ручного годування, ввімкнення режиму доналаштування. Індикатор потрібен, оскільки в системі відсутній будь-який дисплей, тому про певні стани системи необхідно сповіщати користувача.

9.6 Висновки

Оскільки система складається з двох підсистем: пристрою та серверної частини, то й зручний інтерфейс користувача повинен бути присутній в обох підсистемах.

На пристрої інтерфейс користувача представлений, кнопками, світлодіодним індикатором, динаміком.

Серверна ж частина дає можливість взаємодії з системою посередництвом платформи Telegram ботів. У розділі було розглянуто реалізацію дизайну діалогів користувача з Telegram ботом «@alicepetfeeder», що є прототипом інтерфейсу користувача для даної системи. В даного бота є широка функціональність по взаємодії

із клієнтом з допомогою команд, повідомлень та кнопок розмітки.

Було розглянуті необхідні команди для бота, щоб забезпечував обрану функціональність. В основному команди відповідають сценаріям використання. Мова спілкування та команд – англійська, тому що вона є інтернаціональною. Та в майбутньому можливо буде зробити налаштування мови, якщо це матиме бізнес-значимість.

10 СТАРТАП

10.1 Опис ідеї проекту

Проект: підприємство, що виготовлятиме пристрої для годування домашніх тварин. Особливостями буде можливість віддаленого керування, інтеграція з Telegram, фіксація фото та керування кількістю корму.

Таблиця 10.1 – Опис ідеї стартап-проекту

Зміст ідеї	Напрямки використання	Вигоди для користувача
Система автоматичного годування тварин на базі міні-комп'ютера	Встановлення вдома	Автоматичність процесу годування домашніх тварин дозволить економити час та зусилля. Встановлення розміру порції допоможе контролювати здорове харчування домашнього улюбленця. Нагадування про те, що корм закінчується.

Основними техніко-економічними характеристиками ідеї є:

- автоматичне годування тварини;
- захищеність зв'язку;
- віддалене керування пристроєм;
- широка функціональність по налаштуванню процесу годування;
- можливість налаштування розміру порції;
- доступ до системи онлайн;
- можливість зробити фото тварини;

- перегляд журнал годування;
- нагадування про закінчення корму.

Є годівниці різного класу серед конкурентів (див. розділ 1.1). Серед найближчих по функціональності конкурентів можна зазначити годівниці PetWant PF-103, Petkit Smart Feeder, Smart feed від PetSafe. Для визначення сильних, нейтральних та слабких характеристик системи, було здійснено збір та аналіз інформації щодо значень техніко-економічних показників для ідеї власного проекту та проектів-конкурентів відповідно до визначеного вище переліку. Таблиця 10.2 містить порівняння даної системи із конкурентами, в ній стовпчик W – слабка сторона, N - нейтральна сторона, S - сильна сторона.

Таблиця 10.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проекту.

№	Техніко-економічні характеристики ідеї	Потенційні товари/концепції конкурентів				W	N	S
		Мій проект	PetWant	Petsafe	Petkit			
1.	Автоматичне годування тварини	Має	Має	Має	Має	-	+	-
2.	Віддалене керування пристроєм	Має	Немає	Має	Має	-	-	+
3.	Широка функціональність по налаштуванню процесу годування	Має	Немає	Немає	Немає	-	-	+
4.	Можливість налаштування розміру порції	Має	Має	Має	Має	-	+	-
5.	Доступ до системи	Має	Немає	Немає	Немає	-	+	-

	онлайн							
6.	Можливість зробити фото тварини	Має	Має	Немає	Немає	-	-	+
7.	Перегляд журналу годування	Має	Немає	Немає	Немає	-	-	+
8.	Нагадування про закінчення корму	Має	Немає	Має	Немає	-	-	+
9.	Автоматичне замовлення нової пачки корму	Немає	Немає	Немає	Має	+	-	-

10.2 Технологічний аудит ідеї проекту

Для вдалої реалізації проекту, необхідно провести огляд існуючих технологій та аналіз на степінь доступності, готовності до використання. По результатам цього, був обраний необхідний набір технологій, встановлена їх доступність. Таким чином була встановлена можливість технологічної здійсненності ідеї проекту, і як підсумок оформлена у таблиці 10.3

Таблиця 10.3 – Технологічна здійсненність ідеї проекту.

№	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1.	Керування автоматичною видачею порції корму	Мінікомп'ютер Raspberry Pi	Наявна	Доступна
2.	Механічна видача корму	Сервоприводи	Наявна	Доступна
3.	Віддалене керування пристроєм	Wifi, Internet	Наявна	Доступна
4.	Фотозйомка	Цифрова камера,	Можливо	Доступна

		драйвера для Raspberry pi	доведеться доробити	
5.	Реалізація бізнес-логіки та користувацького інтерфейсу	Веб-фреймворк ASP.NET Core, мова програмування C#	Наявна	Доступна
6.	Інтерфейс керування через Telegram	Додаток Telegram, Telegram API	Наявна	Доступна
7.	Серверна частина	ASP.NET Core	Наявна	Доступна
8.	Безпечний зв'язок	Протокол HTTPS	Наявна	Доступна
9.	Збереження даних сервері	Microsoft SQL Server	Наявна	Доступна
Обрана технологія реалізації ідеї проекту: є можливою.				

Згідно з проведеними дослідженнями, перераховані технології легко доступні до використання та можливостей побудови систем, методи реалізації є доступними, деякі з технологій є платними.

10.3 Аналіз ринкових можливостей запуску стартап-проекту

Визначення ринкових можливостей, які можна використати під час ринкового впровадження проекту, та ринкових загроз, які можуть перешкодити реалізації проекту, дозволяє спланувати напрями розвитку проекту із урахуванням стану ринкового середовища, потреб потенційних клієнтів та пропозицій проектів-конкурентів. Для запуску стартап-проекту у ринок, необхідно визначити стан ринку, визначити його загрози, спланувати напрями розвитку, потреби потенційних клієнтів та пропозицій конкурентів. Попередня характеристика потенційного ринку зображена на таблиці 10.4.

Оскільки демографія зростає, зростають доходи населення, то ринок збільшується. І за результатами проведеного дослідження, для входження ринок є привабливим, а норма рентабельності є задовільною.

Таблиця 10.4 – Попередня характеристика потенційного ринку стартап-проекту

№	Показники ринку (найменування)	Характеристика
1.	Кількість головних гравців, од	1 (розробники)
2.	Загальний обсяг продаж, грн/ум. од.	Понад 2000
3.	Динаміка ринку (якісна ціна)	Зростає
4.	Наявність обмежень для входу	Конкуренція
5.	Специфічні вимоги до стандартизації та сертифікації	Відсутні
6.	Середня норма рентабельності в галузі (або по ринку), %	130%

Надалі, необхідно визначити потенційних клієнтів, їх характеристики та сформувані перелік вимог до кожного виду товару. Характеристика описана в таблиці 10.5.

Таблиця 10.5 – Характеристика потенційних клієнтів стартап-проекту

Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
Потреба в економії часу та зусиль клієнтів щодо годування домашніх тварин	Середній та заможний класи населення з міста, що мають домашніх тварин	Поведінка клієнта формується від його потреби в економії часу та зусиль на запам'ятовування.	- висока якість пристрою - легкість у використанні - приємна ціна - надійність роботи - зручність використання

Внаслідок проведення аналізу стосовно потенційних груп клієнтів, було встановлено, що основна частина клієнтів знаходиться серед заможного та середнього класів населення. Тому зосередитись важливо на їх основних вимогах, в першу чергу надійність роботи пристрою та зручність використання. Після визначення вимог клієнтів необхідно провести огляд ринкового середовища, розглянути фактори, які сприяють впровадженню проекту на ринку, та фактори, що йому перешкоджають (табл. 10.6, табл. 10.7).

Таблиця 10.6 – Фактори загроз стартап-проекту

№	Фактор	Зміст загрози	Можлива реакція компанії
1.	Проблеми з постачальниками	Є сильна залежність від постачальників складових даної системи	Пошук інших постачальників, впровадження альтернативних компонентів
2.	Виявлення нових технологій в наслідку наукового прориву	Поява кращих технологій, компонентів	Використання нових технологій чи компонентів у нових моделях/продуктах компанії
3.	Поява нової системи-конкурента на ринку	Поява конкурентів	Докладний аналіз продукту конкурента, адаптація проекту
4.	Складність у використанні	Програмний продукт виявляється складним для використання старшим поколінням	Орієнтація на більш молодих людей чи спрощення використання

5.	Мала кількість продажів	Прогнозовані продажі не виправдались	Пошук нових можливостей покращення продукту, збирання та дослідження відгуків користувачів
6.	Світова криза	Люди починають витратити менше грошей, і дана система матиме низьку пріоритетність	Розробка та виготовлення пристроїв у різних галузях чи напрямках застосування для диверсифікації ризиків
7.	Психологічний фактор	Деяким клієнтам, можливо, з психологічної точки зору, важливо власноруч годувати своїх домашніх тварин.	Встановлення кнопки годування на пристрої

Таблиця 10.7 – Фактори можливостей стартап-проекту

№	Фактор	Зміст можливості	Можлива реакція компанії
1.	Зацікавленість в продукті іноземних клієнтів	Розширення ринку, нові клієнти	Дороблення системи для задоволення вимог нових клієнтів
2.	Зацікавленість у продукті спонсорів або інвесторів	Збільшення капіталу проекту	Масштабування продукту, збільшення маркетингу
3.	Ріст кількості людей із домашніми тваринами	Розширення модельного ряду, ринку, нові клієнти	Запровадження модифікацій системи в залежності від видів тварин

Для проекту системи годування тварин були встановлені основні фактори загроз і можливостей. Загрози є різноманітного плану і пов'язані як з конкурентами так із постачальниками чи пов'язані з появою нових або розвитком старих технологій, що

сприяє появі нових конкурентів і змушує проект адаптуватися. Все ж, було встановлено, що загрози не є дуже критичними, а шанси їх виникнення є доволі низькими. Можливості же сприяють ринковому впровадженню, призводять до зростання попиту і відповідно масштабування проекту. Над ризиками загроз можливості позитивні переважають. Таким чином, проект має потенціал розвитку в ринковому середовищі та можливості для росту, а значить необхідно визначити загальні риси конкуренції на ринку (табл. 10.8).

Таблиця 10.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив діяльності підприємства
Тип конкуренції: чиста	Конкурентами є компанії із приблизно однаковою долею ринку	Виготовлення продукту вищої якості, зручнішим функціоналом, нижчою ціною
За рівнем конкурентної боротьби: світовий	Ринок охоплює всю планету	Забезпечення відповідності продукту стандартам різних країн
За галузевою ознакою: внутрішньо-галузева	Конкуренти знаходяться в одній галузі (розумна електроніка для дому)	Аналіз факторів, які впливають на успіх у даній галузі
Конкуренція за видами товарів: товарно-родова, товарно-видова	Товар конкурентів як одного виду – розумні годівниці, так і інші типи годівниць	Продукт має бути кращим у галузі
За характером конкурентних переваг: не цінова	Товар якісніший за продукти конкурентів	Вкладання зусиль, ресурсів у підтримання вищої якості продукту

За інтенсивністю: не марочна	Товар тільки виходить на ринок	Забезпечити успішний старт продукту на ринку
------------------------------	--------------------------------	--

Завдяки ступеневому аналізу конкуренції на ринку, можна зробити висновок, що конкурентні товари, хоч і утримують позиції присутні на ринку, та ринок ще перспективний. Товари конкурентів мають значні недоліки, які проект може використати для укріплення позицій на ринку. Більш детальний аналіз умов конкуренції в галузі - далі (табл. 10.9).

Таблиця 10.9 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	Розумні годівниці PetWant, Petkit, PetSafe	Конкурентом може стати будь-який продукт що вирішуватиме задачу автоматичного годування тварин	Деякі постачальники мають монополію	Середній та заможній класи населення	Більш дешеві китайські версії
Висновки	Інтенсивність конкурентної боротьби доволі низка	Потенційні конкуренти можуть виникнути згодом	Існує сильна залежність від постачальників, проте вони не диктують умови роботи ринку.	Головна вимога клієнтів це надійність роботи	Існує фактор ризику товарів замінників

Аналіз за М. Портером показав, що наявні декілька прямих конкурентів та є загроза товарів-замінників. Але ситуація сприятлива і інтенсивність конкурентної боротьби низька. Головною вимогою клієнтів є надійність роботи. Хоча й існує сильна залежність від постачальників але все ж проект може мати успіх. Проте, для цього необхідно встановити ключові сильні сторони, які повинен мати проект, щоби бути конкурентоспроможним на ринку. Обґрунтування факторів конкурентоспроможності описано в таблиці 10.10.

Таблиця 10.10 – Обґрунтування факторів конкурентоспроможності

№	Фактор конкурентоспроможності	Обґрунтування факторів
1.	Віддалене керування	Продукт дає можливість керування з будь-якої точки Інтернету
2.	Зручність використання	Продукт економить час клієнта, автоматизуючи процес годування домашньої тварини
3.	Комплексний підхід	Поєднання усього функціоналу в одному місці покращує взаємодію користувача з системою
4.	Можливість розширення продукту	Продукт передбачає можливість розширення, шляхом створення спеціалізованих дієт для певних типів тварин
5.	Наявність камери	Використання камери дозволить перевірити чи домашній улюбленець поїв, чи зробити фото, чи в майбутньому налаштувати відео дзвінок
6.	Інтеграція з месенджером Telegram	Зручність та простота використання

Щоб гарантовано мати успіх у сучасній ринковій ситуації, для проекту були обрані фактори, на яких буде заснована його перевага над конкурентами. Найважливішими серед факторів конкурентоспроможності є віддалене керування,

інтеграція із месенджером Telegram, можливості розширення продукту шляхом створення спеціалізованих підходів годування для різних тварин. Порівняння факторів конкурентоспроможності дає можливість визначити сильні та слабкі сторони стартап-проекту (табл. 10.11).

Таблиця 10.11 – Порівняльний аналіз сильних і слабких сторін стартап-проекту

Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів порівняно з проектом						
		-3	-2	-1	0	+1	+2	+3
Віддалене керування	6		PetKit	PetSafe	PetWant			
Зручність використання	10			Petwant PetSafe		Petkit		
Можливість розширення продукту	5				Petkit	Petwant PetSafe		
Наявність камери	10		Petkit		Petwant PetSafe			
Інтеграція з месенджером Telegram	18		Petkit Petwant PetSafe					

Порівняльний аналіз продемонстрував, що за факторами конкурентоспроможності, значні переваги над конкурентами має стартап-проект і тому може мати успіх на ринку. Для повноти аналізу, необхідно складання SWOT-аналізу на основі ринкових можливостей та загроз, а також сильних і слабких сторін (табл 10.12).

Таблиця 10.12 – SWOT-аналіз стартап-проекту

Сильні сторони: - актуальність проекту;	Слабкі сторони: - не найсприятливіше ринкове
--	---

<ul style="list-style-type: none"> - функції, що відсутні у конкурентів; - зручне віддалене керування; - наявність потреби у економії часу; - потреба користувачів у слідкуванні за тим як харчуються їх тварини. 	<p>середовище;</p> <ul style="list-style-type: none"> - складність масштабного впровадження; - сильна залежність від постачальників.
<p>Можливості:</p> <ul style="list-style-type: none"> - поява інвесторів; - розширення ринку в наслідок зацікавленості в продукті; - послаблення позицій конкурентів; - поява проривних технологій. 	<p>Загрози:</p> <ul style="list-style-type: none"> - проблеми з постачальниками; - поява нової системи-конкурента на ринку; - мала кількість продажів;

Як наслідок проведення SWOT-аналізу, були розроблені альтернативи ринкової поведінки для виведення проекту системи автоматичного годування на ринок (табл. 10.13). SWOT-аналіз дав змогу оцінити з різних точок і порівняти ключові сторони проекту та ринкового середовища.

Таблиця 10.13 – Альтернативи ринкового впровадження

Альтернатива (орієнтований комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
Загарбник	Середня	Більше 2 років
Наступник	Суттєва	1-2 роки
Виклик лідеру	Середня	1-2 роки

Для стартап-проекту були обрані три альтернативи ринкової поведінки – загарбник, наступник, виклик лідеру. Використання альтернативи Наступник дозволить спеціалізуватися на одній частині ринку, а потім поглиблювати свій вплив. Виклик лідеру має середню ймовірність отримання ресурсів. Іншою альтернативою є загарбник, який орієнтований на захоплення ринку від конкурентів. Але найкраще за

строком реалізації та ймовірністю отримання ресурсів підходить наступник.

10.4 Розроблення ринкової стратегії проекту

Для реалізації проекту на ринку, необхідно розробити положення, що коротко описують потенційних споживачів, цільовий ринок, спосіб позиціонування товару і величини обсягів продажу, яких планується досягнути за перші кілька років реалізації товару. Для цього зробимо перший крок визначення стратегії охоплення ринку: опис цільових груп потенційних споживачів (табл. 10.14).

Таблиця 10.14 – Вибір цільових груп потенційних споживачів

Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтований попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу в сегмент
Заможний клас населення	Висока	Високий	Висока	Середня
Середній клас	Середня	Середній	Низька	Середня
Які цільові групи обрано: середній та заможний класи населення				

За результатами аналізу потенційних груп споживачів (сегментів) можна сказати, що цільовими групами, що матимуть мотивацію та змогу купити даний продукт є середній та заможний класи населення. Оскільки дані групи складають більшу частину населення, то пропонується використовувати стандартизовану програму масового маркетингу.

Наступний крок розробки ринкової стратегії полягає в визначення базової стратегії розвитку (табл. 10.15).

Таблиця 10.15 – Визначення базової стратегії розвитку

Обрана	Стратегія	Ключові	Базова
--------	-----------	---------	--------

альтернатива розвитку проекту	охоплення ринку	конкурентоспроможні позиції відповідно до обраної альтернативи	стратегія розвитку
Виклик лідеру	Концентрація на потребах великого сегменту ринку	Надання клієнтам сучасного програмно-апаратного продукту з певним унікальним функціоналом	Стратегія диференціації

З відомих стратегій розвитку для першого періоду стартапу була обрана стратегія диференціації, як найбільш підходяща. Ця стратегія передбачає надання товару важливих з точки зору споживача відмітних властивостей, які роблять товар відмінним від товарів конкурентів. Стартап-проект має на меті випускати продукт кращої якості з унікальним функціоналом. Тому обрана стратегія є підходящою. Наступним пунктом в розробці ринкової стратегії полягає у виборі стратегії конкурентної поведінки (табл. 10.16).

Таблиця 10.16 – Визначення базової стратегії конкурентної поведінки

Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
Проект є наступником вже існуючих товару	Перший час проект буде зваблювати нових споживачів, а згодом переманювати	Необхідно реалізувати такі характеристики конкурентів: - автоматичне годування; - зручність використання; - надійність	Стратегія виклику лідера.

	існуючих у конкурентів		
--	------------------------	--	--

Отже, як базову стратегію конкурентної поведінки було обрано стратегію виклику лідеру. З наявних, вона найкраще відповідає цілям проекту і полягає в активній конкуренції з лідерами ринку з метою посунути їх з лідерської позиції. Наступним кроком є визначення стратегії позиціонування (табл. 10.17).

Таблиця 10.17 – Визначення стратегії позиціонування

Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
Надійне виконання основної функції годування, зручність інтерфейсу, збереження даних, віддалене керування	Стратегія диференціації	Стратегія виклику лідера	Економія часу, легкість використання, нагадування якщо корм закінчується

Таким чином, позиціонування продукту було визначено на основі SWOT-аналізу, вибору цільових груп потенційних споживачів, визначення стратегії захоплення ринку та конкурентоспроможних переваг продукту.

10.5 Розроблення маркетингової програми стартап-проекту

Для своєчасної та успішної реалізації проекту, треба розробити програму

маркетингу, систему взаємозалежних заходів, що визначають дії підприємства-виробника на заданий період часу з усіх питань маркетингової діяльності. Формування програм маркетингу відбувається на підставі даних щодо комплексного дослідження ринку, визначення поточних і перспективних потреб і попиту потенційних споживачів, з урахуванням обраної стратегії і тактики маркетингу[48]. Першим головним кроком є формування маркетингової концепції товару, який отримає клієнт. Необхідно здійснити визначення ключових переваг концепції потенційного товару (табл. 10.18).

Таблиця 10.18 – Визначення ключових переваг концепції потенційного товару

Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
Потреба в економії часу	Зручний у використанні сучасний програмно-апаратний продукт, який має певні унікальні переваги над аналогами	Ключовими перевагами є: нагадування про закінчення корму, інтеграція з Telegram
Потреба в здоровому харчуванні (дієті) для домашніх тварин	Зможе слідкувати за вагою, а отже здоров'ям тварини	Можливість створення графіків годування різного виду, графік годування – дієта.

Ключовими перевагами концепції даного проекту є нагадування про закінчення корму, інтеграція з Telegram, тобто інтерфейс користувача, можливість створення графіків годування різного виду, графік годування – дієта. Подальшим кроком є розробка трирівневої маркетингової моделі товару, для уточнення ідеї продукту або послуги, особливості процесу надання, його фізичні складові (табл. 10.19).

Таблиця 10.19 – Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові		
	Опис базової потреби споживача, яку задовольняє товар (згідно концепції), її основної функціональної вигоди		
	Продаж пристроїв для автоматичного годування тварин, що дозволить економити час їх власникам		
	Властивості/ характеристики	М/Нм	Вр/Тх/Тл/Е/ Ор
	<p>1. Економічні: вартість обслуговування та ремонту планується зменшити за рахунок якості товару</p> <p>2. Технологічні: використання сучасних технологій</p> <p>3. Ергономічність: зручний користувацький інтерфейс, зрозумілий функціонал продукту</p> <p>4. Естетичні: якісний гарний дизайн продукту</p> <p>5. Безпеки: відповідність нормативам використання електронних пристроїв та програмного забезпечення</p> <p>6. Надійності: система годування повинна справно працювати по проходженню декількох років, гарантія в 5 років</p> <p>7. Транспортабельності: пристрій транспортабельний, може бути переданим від одного споживача іншому</p> <p>8. Екологічні: відповідність нормативам використання електронних пристроїв та програмного забезпечення</p>	-/+	+//+//+//+
	Якість: нормативи безпеки електронних пристроїв		

	Дизайнерське пакування,
	Марка: Alice Petfeeder
	До продажу: представлення клієнтам продукту
	Після продажу: допомога в налаштуванні, ремонт, підтримка продукту
За рахунок чого потенційний товар буде захищено від копіювання: неможливості реверс інжинірінгу програмного коду, захищеність операційної системи на борту пристрою	

Наступним кроком є визначення цінових меж, якими необхідно керуватись при встановленні ціни на потенційний товар, яке передбачає аналіз ціни на товари-аналоги або товари субституту, а також аналіз рівня доходів цільової групи споживачів (табл. 10.20).

Таблиця 10.20 – Визначення меж встановлення ціни

Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
100-200 ум. од.	100-200 ум. од.	400, 1000+ ум. од.	90-190 ум. од.

Оскільки цільових груп споживачів – дві, але одна значно більша за іншу, то при встановленні цін, варто керуватися можливостями більшої групи. Орієнтовна ціна націлена на можливості середнього класу населення та є нижчого ніж в аналогів. Таким чином, потенційний товар є доступним для всіх цільових груп споживачів. Наступним кроком є формування системи збуту (10.21).

Таблиця 10.21 – Формування системи збуту

Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
Покупка електронних пристроїв спеціалізованих магазинах	Доставка, надання консультацій при продажі та використанні	Глибока	Сторонні посередники
Покупка електронних пристроїв в Інтернеті	Інформування потенційних споживачів, замовлення товарів через сайт	Мінімальна	Власні сили, сторонні майданчики

Для оптимальності системи збуту було вирішено проводити продаж власними силами, використовувати інтернет-майданчики для розповсюдження товару та співпрацювати з магазинами електронної техніки. Останнім кроком маркетингової програми є розроблення маркетингових комунікацій (табл. 10.22).

Таблиця 10.22 – Концепція маркетингових комунікацій

Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
Серед клієнтів багато людей, що проводять багато часу в Інтернеті та	Соціальні мережі, Інтернет, Сарафанне радіо	Сучасний продукт, заощаджує час, легкість у використанні,	Донести користувачам інформацію про якісний і сучасний	Оригінально сповістити споживачів про новий продукт

соціальних мережах		дбає про вашого улюбленця	продукт з перевагами	
--------------------	--	---------------------------	----------------------	--

Основними каналами комунікаціями для користувачів, є Інтернет і соціальні мережі, і використання таких каналів є необхідним. Маркетингова кампанія в мережі дозволяє охопити більш широку аудиторію і зацікавити більше клієнтів. Ключовими позиціями продукту є його сучасність та певний унікальний функціонал.

10.6 Висновки до розділу

Проект під маркою Alice Petfeeder є системою автоматичного годування тварин на базі міні-комп'ютера. Автоматичність процесу годування домашніх тварин дозволить економити час та зусилля клієнтів. Встановлення розміру порції допоможе контролювати здорове харчування домашнього улюбленця та влаштувати дієту тварині. Нагадування про те, що корм закінчується дасть можливість користувачам розвантажити свідомість від запам'ятовування такої інформації. А для компанії потенційну можливість реклами корму.

На ринку представлено декілька найближчих по функціональності конкурентів. Було визначено сильні, нейтральних та слабкі характеристик системи в порівнянні з ними. Порівняння виявило, що проект є перспективним. Технічний аудит проекту було проведено і виявлено, що технології для реалізації ідеї наявні на ринку та доступні для впровадження.

За результатами дослідження потенційного ринку проведеного дослідження, для системи годування тварин входження ринок є привабливим, а норма рентабельності є задовільною.

Аналіз потенційних клієнтів показав, що середній та заможний класи населення з міста, що мають домашніх тварин є цільовими категоріями споживачів для проекту. Їх головними вимогами є заощадження часу та зручність користування. Система, за задумкою, успішно задовольняє ці вимоги.

Для реалізації проекту існують загрози різноманітного плану і пов'язані як з

конкурентами так із постачальниками, чи пов'язані з появою нових або розвитком старих технологій, що сприяє появі нових конкурентів і змушує проект адаптуватися. Але було встановлено, що загрози не є надто критичними, а шанси їх виникнення є не настільки значними. Можливості же сприяють ринковому впровадженню, призводять до зростання попиту і відповідно масштабування проекту. А щоб гарантовано мати успіх у сучасній ринковій ситуації, для проекту були обрані фактори, на яких буде заснована його перевага над конкурентами. Найважливішими серед факторів конкурентоспроможності є віддалене керування, інтеграція із месенджером Telegram, можливість розширення продукту шляхом створення спеціалізованих підходів годування для різних тварин.

Проект має перспективи для масштабування так як потенційний ринок охоплює всю планету. Тому з відомих стратегій розвитку для першого періоду стартапу була обрана стратегія диференціації, як найбільш підходяща. А згодом планується проводити обробку даних продажів і планувати подальшу стратегію розвитку.

Маркетингова програма продукту дозволить своєчасно та успішно реалізувати проект. У маркетинговій програмі продукту було використано ключові переваги концепції даного проекту, а саме нагадування про закінчення корму, інтеграція з Telegram, тобто зручний та сучасний інтерфейс користувача, можливість створення графіків годування різного виду, графік годування – дієта. Позиціонування товару буде засноване на його ключових відмінних характеристиках

Встановлені цінові межі були обґрунтованими для цільових груп клієнтів та можливі для досягнення. Система збуту товару складатиметься з двох складових: Інтернет-продажів та продажів у спеціалізованих магазинах електронної техніки.

Оптимальними каналами для комунікації зі споживачами орано Інтернет, та соціальні мережі зокрема.

ВИСНОВКИ

В результаті роботи над магістерською дисертацією було створено систему годування тварин на базі мінікомп'ютера. Розроблена система вирішує проблему економії часу для власників тварин, та проблему здорового харчування для самих тварин.

При розробці системи, було проаналізовано схожі продукти. Розглянуто автоматичні годівниці для тварин: PetWant PF-103, Petkit Smart Feeder, Pet safe. Після аналізу були визначені проблеми, що лягли в основу даної роботи. Було визначено функції які ще не реалізовані і які потенційно можуть мати попит.

Внаслідок аналізу існуючих рішень було сформовано вимоги до системи. Головними функціональними вимогами є автоматичне годування по графіку, можливості створення графіків годування та їх зміни, видалення, взаємодія з користувачем через Telegram, надсилання сповіщень про закінчення корму, перегляд статистики годування. Серед нефункціональних вимог можна виділити захищеність комунікації.

Перед реалізацією системи було визначено сценарії використання. Головним актором є користувач, який по задумці купує пристрій. Його сценарії використання в основному відображають вимоги системи. Другим актором є адміністратор, його роль у внесенні певних даних про пристрої у систему та створення дієт для тварин. В даній роботі розглянуто реалізацію сценаріїв використання лише користувача.

У складі система годування дві головні підсистеми - це пристрій та серверна частина. Для реалізації було проаналізовано існуючі компоненти та модулі для створення обох підсистем. Серверну частину було вирішено робити на базі веб-фреймворку ASP.NET Core і мові програмування C#. При цьому сховищем даних обрано MSSQL. Для використання Telegram API було вирішено використати готовий пакет Telegram.Bots.Framework, що пришвидшив розробку та дозволив абстрагуватися від дрібних деталей реалізації зв'язку з Telegram. Зокрема, було використано Nuget пакети FluentScheduler та NCrontab.Advanced. Для апаратної частини було обрано RaspberryPi із Raspbian OS, сервоприводи та I2C модуль

курування сервоприводами PCA9685. Апаратну частину було вирішено запрограмувати за допомогою мови програмування Python і використовувати готові пакети для нього.

Реалізована підсистема пристрою структурно складається із таких компонентів мінікомп'ютер, кнопки та перемикач, сервоприводи, динамік, цифрова камера, модуль керування сервоприводами. Виконавчими механізмами, що здійснюють основну функцію системи («видати корм») є два сервоприводи. Перший для дозування корму, інший – для відкривання-закривання засуву, що має забезпечити корм від неконтрольованого поїдання.

Процес взаємодії між пристроєм та сервером і між сервером та Telegram відбувається через захищений протокол HTTPS. Пристрій повинен бути під'єднаний до Інтернету для того, щоб користувач мав змогу взаємодіяти з ним через Telegram. Обмін даними відбувається в режимі опитування - пристрій періодично опитує сервер на оновлення даних чи надходження нових команд. В свою чергу сервер отримує команди від Telegram і зберігає, поки їх не буде виконано на пристрої.

Для реляційного сховища даних було здійснено аналіз доменної області, визначення сутностей, що відображено на ER-діаграмі та розроблено схему БД. Основні сутності, що зберігаються у БД є: користувач, пристрій, графік годування, рядок часу, доступ до пристрою, журнал годувань. Варто виділити сутність рядок часу, так як графік годування по суті складається із рядків часу. Кожен рядок часу описує один момент дня з допомогою виразу `stop`, саме цей момент і є запланованим годуванням.

За основу архітектури серверної частини було взято багат шарову цибулеву архітектуру. В окремому модулі було визначено основну бізнес-логіку, в окремому модулі оголошені інтерфейси та спільні класи, що використовується в інших модулях. Конкретні реалізації інтерфейсів були рознесені по модулям відповідно до шарів. А для зменшення зв'язності використано паттерн Інверсія залежностей. Процес обробки запитів сервером був реалізований з допомогою паттерну CQRS та бібліотеки MediatR.

Для системи було розроблено новий для таких систем інтерфейс користувача

через Telegram бота. Забезпечена обробка команди та кнопок в інтерфейсі, що відповідають сценаріям використання системи. Зокрема додано зручну для користувача довідку команд.

Потенціал системи було розглянуто в розрізі стартапу та визначено, що є перспективи у цього проекту і шанс вийти на ринок. Потенційними покупцями даного пристрою є заможний та середній клас міст. Здійснено технологічний аудит проект, аналіз сильних та слабких сторін продукту, ринкових загроз та можливостей, визначено стратегії позиціонування на ринку та розроблено на основі цього програму маркетингу.

В подальшому проект можна розвивати, оптимізувати та покращувати. Пристрій потрібно здешевлювати. У роботі було описано та не реалізовано сценарій першого налаштування системи, під час якого користувач підключається через Bluetooth до мінікомп'ютера та встановлює пароль для локальної бездротової мережі. Для повноцінного впровадження системи це потребує реалізації. Мінікомп'ютер потенційно можна винести за рамки пристрою і зробити центром системи розумних пристроїв для будинку. Щодо серверної частини, то в роботі не було розглянуто аспекти масштабування системи. Зокрема потрібно провести ряд тестувань системи.

В результаті проектування створено макет пристрою та Telegram бот.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Кормушка для животных. Для чего нужна и как выбрать [Электроний ресурс] – Доступ: https://media.price.ua/gid_pokupatelya/avtomaticheskaya-kormushka-dlya-zhivotnyh-dlya-chego-nuzhna-i-kak-vybrat.html
2. Healthy Pet Simply Feed [Электроний ресурс] – Доступ: <https://store.petsafe.net/healthy-pet-simply-feed>
3. Petwant [Электроний ресурс] – Доступ: <https://petwant.org/>
4. Petkit Smart Feeder [Электроний ресурс] – Доступ: <https://www.petkit.com/products/petkit-smart-feeder-automatic-pet-feeder-for-cats-and-dogs-a-never-stuck-feeder-double-fresh-lock-system-5-9l-large-capacity>
5. Smart Feed Original [Электроний ресурс] – Доступ: <https://store.petsafe.net/smart-feed-original>
6. Amazon Echo [Электроний ресурс] – Доступ: https://ru.wikipedia.org/wiki/Amazon_Echo
7. Amazon Alexa [Электроний ресурс] – Доступ: https://uk.wikipedia.org/wiki/Amazon_Alexa
8. Пять секретов правильного питания кошки [Электроний ресурс] – Доступ: <https://www.crimea.kp.ru/daily/24049.4/102475/>
9. Telegram FAQ [Электроний ресурс] – Доступ: <https://telegram.org/faq>
10. Viber [Электроний ресурс] – Доступ: <https://www.viber.com/>
11. Bots: An introduction for developers [Электроний ресурс] – Доступ: <https://core.telegram.org/bots>
12. Viber API Documentation [Электроний ресурс] – Доступ: <https://developers.viber.com/docs/api/rest-bot-api/>
13. Самые популярные мессенджеры в Украине [Электроний ресурс] – Доступ: <https://delo.ua/lifestyle/nazvany-samyepopuljarnyemessendzheryv-ukraine-360645/>
14. Что не так с Raspberry Pi [Электроний ресурс] – Доступ: <https://habr.com/ru/post/440584/>
15. Arduino [Электроний ресурс] – Доступ: <https://uk.wikipedia.org/wiki/Arduino>

16. Arm Processors for the Widest Range of Devices—from Sensors to Servers [Электроний ресурс] – Доступ: <http://www.arm.com/products/processors/cortex-m/index.php>

17. Сервопривод [Электроний ресурс] – Доступ: <https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%BE%D0%BF%D1%80%D0%B8%D0%B2%D0%BE%D0%B4>

18. Сервоприводы. Виды и устройство. Характеристики и применение [Электроний ресурс] – Доступ: <https://electrosam.ru/glavnaja/slabotochnye-seti/oborudovanie/servoprivody/>

19. Adafruit 16-Channel 12-bit PWM/Servo Driver - I2C interface - PCA9685 [Электроний ресурс] – Доступ: <https://www.adafruit.com/product/815>

20. Build secure IoT devices with Ubuntu Core [Электроний ресурс] – Доступ: <https://ubuntu.com/download/raspberry-pi>

21. Fedore IoT [Электроний ресурс] – Доступ: <https://iot.fedoraproject.org/>

22. Raspbian [Электроний ресурс] – Доступ: <https://ru.wikipedia.org/wiki/Raspbian>

23. Python [Электроний ресурс] – Доступ: <https://ru.wikipedia.org/wiki/Python>

24. Crontab package [Электроний ресурс] – Доступ: <https://pypi.org/project/crontab/>

25. Requests package [Электроний ресурс] – Доступ: <https://pypi.org/project/requests/>

26. Introduction to ASP.NET Core [Электроний ресурс] – Доступ: <https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-3.0>

27. Nuget package manager [Электроний ресурс] – Доступ: <https://en.wikipedia.org/wiki/NuGet>

28. GPIO в Raspberry Pi, эксперимент со светодиодом и кнопкой [Электроний ресурс] – Доступ: <https://ph0en1x.net/86-raspberry-pi-znakomstvo-s-gpio-perekluchatel-i-svetodiod.html>

29. Opencv-python package [Электроний ресурс] – Доступ: <https://pypi.org/project/opencv-python/>

30. Adafruit-circuitpython-servokit package [Электроний ресурс] – Доступ: <https://pypi.org/project/adafruit-circuitpython-servokit/>

31. Порты общего назначения в Raspberry Pi [Электроний ресурс] – Доступ: <https://ph0en1x.net/86-raspberry-pi-znakomstvo-s-gpio-perekluchatel-i-svetodiod.html>
32. Cron expression [Электроний ресурс] – Доступ: https://en.wikipedia.org/wiki/Cron#CRON_expression
33. Common web application architectures [Электроний ресурс] – Доступ: <https://docs.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/common-web-application-architectures>
34. Spaghetti code [Электроний ресурс] – Доступ: https://en.wikipedia.org/wiki/Spaghetti_code
35. The Principles of OOD [Электроний ресурс] – Доступ: <http://butunclebob.com/ArticleS.UncleBob.PrinciplesOfOod>
36. Software Design Principles DRY and KISS [Электроний ресурс] – Доступ: <https://dzone.com/articles/software-design-principles-dry-and-kiss>
37. Test Doubles — Fakes, Mocks and Stubs [Электроний ресурс] – Доступ: <https://blog.pragmatists.com/test-doubles-fakes-mocks-and-stubs-1a7491dfa3da>
38. The Clean Architecture [Электроний ресурс] – Доступ: <https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html>
39. The Onion Architecture [Электроний ресурс] – Доступ: <https://jeffreypalermo.com/2008/07/the-onion-architecture-part-1/>
40. Clean Architecture with ASP.NET Core 2.2 [Электроний ресурс] – Доступ: <https://www.youtube.com/watch?v=Zygw4UAxCdg>
41. IoC, DI, IoC-контейнер [Электроний ресурс] – Доступ: <https://habr.com/ru/post/131993/>
42. Command and Query Responsibility Segregation (CQRS) pattern [Электроний ресурс] – Доступ: <https://docs.microsoft.com/en-us/azure/architecture/patterns/cqrs>
43. MediatR library wiki [Электроний ресурс] – Доступ: <https://github.com/jbogard/MediatR/wiki>
44. What is HTTP Long Polling? [Электроний ресурс] – Доступ: <https://www.pubnub.com/blog/http-long-polling/>
45. Webhook [Электроний ресурс] – Доступ:

<https://uk.wikipedia.org/wiki/Webhook>

46. Webhook vs polling [Електроний ресурс] – Доступ:

<https://community.stuart.engineering/t/webhooks-vs-polling/257>

47. Smileys & People [Електроний ресурс] – Доступ: <https://emojipedia.org/people/>

48. Програма маркетингу [Електроний ресурс] – Доступ:

https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B0_%D0%BC%D0%B0%D1%80%D0%BA%D0%B5%D1%82%D0%B8%D0%BD%D0%B3%D1%83