МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ЦЕНТР ЗАОЧНОЇ, ДИСТАНЦІЙНОЇ ТА ВЕЧІРНЬОЇ ФОРМ НАВЧАННЯ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: «Мобільний додаток підтримки діяльності садової теплиці»

за спеціальністю 122 «Комп’ютерні науки»,
освітньо-професійна програма «Інформаційні технології проектування»

Виконавець роботи: студент групи ІТз-81с Горячев Олександр Денисович

Кваліфікаційна робота бакалавра
захищена на засіданні ЕК
з оцінкою _____«__»
2022 р.

Науковий керівник _____к. т. н., доц. Нагорний В. В.
(підпис) (науковий ступінь, вчене звання, прізвище та ініціали)

Засвідчую, що у цій дипломній роботі немає
запозичень з праць інших авторів
без відповідних посилань.
Студент _____(підпис)

Суми-2022

Сумський державний університет

Центр заочної та дистанційної форми навчання

Кафедра інформаційних технологій

Спеціальність 122 «Комп’ютерні науки»

Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ

Зав. кафедри ІТ

_____ В. В. Шендрик

«_____» _____ 2022 р

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА СТУДЕНТУ

Горячев Олександр Денисович

1 Тема роботи *Мобільний додаток підтримки діяльності садової теплиці*

керівник роботи *Нагорний Володимир В’ячеславович, к.т.н., доц,* _____

затверджені наказом по університету від «_»___2022 р.

2 Строк подання студентом роботи «10» червня 2022 р.

3 Вхідні дані до роботи

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) аналіз предметної області, проектування мобільного додатку, розробка мобільного додатку.

5 Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень) сучасний стан проблеми, мета та задачі проекту, аналіз аналогів мобільних додатків содових теплиць, логічна модель даних , функціональні вимоги до мобільного додатку, контекстна діаграма у нотації IDEF0, діаграма декомпозиції першого рівня у нотації IDEF0, діаграма декомпозиції, діаграма

варіантів використання, діаграма потоків даних, діаграма послідовності та діаграма діяльності, засоби реалізації, демонстрація роботи, апробація, висновки.

6. Консультанти розділів роботи:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7.Дата видачі завдання 01.10.2021

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Планування робіт	до 05.04.2022	
2	Розробка технічного завдання	до 11.04.2022	
3	Проведення аналізу предметної області	до 25.04.2022	
4	Проведення структурно-функціонального моделювання	до 5.05.2022	
5	Розробка додатку	до 25.05.2022	
6	Тестування додатку	до 29.05.2022	
7	Оформлення та здача пояснювальної записки та файлів розробленого проекту	до 10.06.2022	

Студент _____ Горячев О.Д.
(підпис)

Керівник роботи _____ к.т.н., доц. Нагорний В. В.
(підпис)

РЕФЕРАТ

Тема кваліфікаційної роботи бакалавра «Мобільний додаток підтримки діяльності садової теплиці».

Пояснювальна записка складається зі вступу, 3 розділів, висновку, списку використаних джерел із 18 найменувань, додатків. Загальний обсяг роботи – 81 сторінок, у тому числі 49 сторінки основного тексту, 2 сторінки списку використаних джерел, 29 сторінка додатків.

Кваліфікаційну роботу бакалавра присвячено розробці мобільного додатку підтримки діяльності садової теплиці.

В першому розділі були розглянуті останні дослідження, проведений аналіз існуючих аналогів додатків теплиць, за допомогою цього визначено переваги та недоліки цих мобільних додатків. З урахуванням недоліків в аналогічних додатках, сформульовані функціональні вимоги ,обрані засоби реалізації, сформульовані мета і задачі.

Потім виконано структурно-функціонального моделювання, моделюванню аналізу мобільного додатку садової теплиці, моделюванню варіантів використання мобільного додатку.

Далі описано процес розробки мобільного додатку, де демонструється його архітектура, апробація та використання, його тестування.

Підсумком кваліфікаційної робота бакалавра є написаний мобільний додаток підтримки діяльності садової теплиці.

Ключові слова: мобільний додаток, Ionic Framework, Angular, *Model-View-Controller*.

ЗМІСТ

ВСТУП.....6
РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ.....8
1.1 Огляд останніх досліджень і публікацій.....8
1.2 Аналіз програмних продуктів – аналогів.....9
1.3 Постановка задання.....16
1.4 Вибір засобів реалізації.....16
2 ПРОЕКТУВАННЯ МОБІЛЬНОГО ДОДАТКУ.....20
2.1 Структурно-функціональне моделювання.....20
2.2 Моделювання варіантів використання мобільного додатку.....23
2.3 Модель аналізу мобільного додатку садової теплиці.....25
3 РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ ПІДТРИМКИ ДІЯЛЬНОСТІ САДОВОЇ ТЕПЛИЦІ.....29
3.1 Архітектура мобільного додатку.....29
3.2 Реалізація мобільного додатку.....31
3.3 Тестування мобільного додатку.....45
3.4 Апробація.....49
ВИСНОВОК.....50
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....52
ДОДАТОК А Технічне завдання.....55
ДОДАТОК Б Планування робіт.....64
ДОДАТОК В Лістинг коду.....77

ВСТУП

Мобільність та швидкий доступ до різноманітної інформації на теперішній час є все більше людей основним. Із року в рік відсоток користувачів мобільних версій сайтів збільшується .

Майже всі програмні продукти розраховані на застосування в мобільних телефонах. Зараз в світі практично не залишилося людей, які б не користувалися мобільним пристроєм. Популярності цього є: функціональність , застосування сучасних засобів зв'язку та невеликого розміру гаджету.

Інформаційні технологій динамічно розвиваються, що відкривають перспективи використання власниками мобільними гаджетами нових сервісів, вчасності збору та обробки інформації щодо стану подій. Дані про події зберігаються в локальних пристроях зберігання даних або в хмарних рішеннях типу спеціалізованих сайтів і баз даних (рис. 1.1).

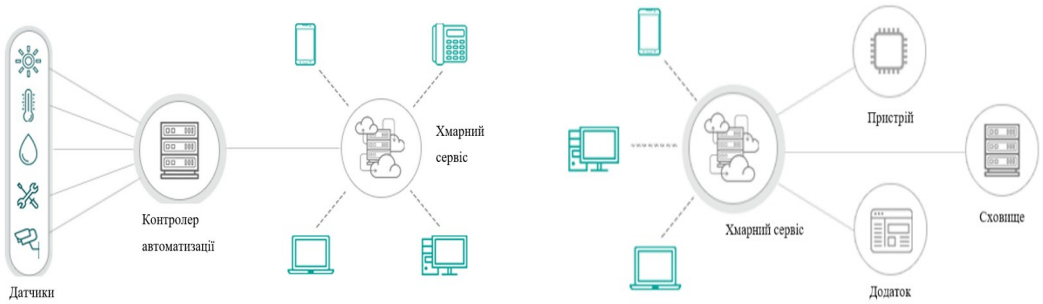


Рисунок 1.1 – Хмарний сервіс інтеграції «розумних» пристроїв

Технічний прогрес не обійшов стороною і садівництво. В першу чергу, процеси автоматизації стали актуальні, звичайно, для поливу рослин, але й інші присадибні завдання тепер можна вирішувати за допомогою техніки.

У зв'язку з цим було вирішено розробити мобільний додаток, який би міг забезпечити швидку та зручну візуалізацію режимів вирощування врожаю в саду (теплиці) шляхом надання інформації з стану виконавчих

механізмів в саду (теплиці), мікроклімату та спеціально створених умов для зрошення врожаю.

Отже, метою роботи є створення мобільного додатку підтримки діяльності садової теплиці.

Для досягнення мети дипломної роботи необхідно буде виконати наступні задачі:

– провести огляд останніх досліджень впровадження інформаційних технологій в сільськогосподарських галузях;

– проаналізувати аналогічні розробки мобільних додатків, пов’язаних з садовими теплицями;

– сформулювати функціональних вимог до мобільного додатку;

– провести структурно-функціональне модулювання та побудувати моделі аналізу мобільного додатку;

– здійснити розробку мобільного додатку садової теплеці;

– провести тестування та апробацію мобільного додатку.

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд останніх досліджень і публікацій

Аналіз літературних джерел [1,2] показали, що аграрії поступово впроваджують інноваційні технології у свої будні.

У виданні «Сучасні проблеми економіки і підприємництва. Випуск 19» [3] автори вважають, що застосування інформаційних технологій в сільськогосподарській галузі, призведе до збільшення продуктивності на обох напрямках. Також вони проводять оцінку ефективності застосування IT у сільській галузі, згадуючи приклади світового досвіду.

Автор американського видання «AgriTechTomorrow» [4] Лен Калдерон підрахував, що фермери усього світу регулярно і активно користуються мобільними додатками.

В основному це – дослідження поля за станом посівів, моніторинг польових робіт, облік і аналіз проблем об’єму врожаю на ньому, контроль за переміщенням сільгосптехніки по полю, тощо.

Завдяки супутниковим знімкам програмний додаток аналізує поля по всій території, попереджаючи аграріїв, коли виникне якась проблема, а саме відстежує інформацію про погоду, моніторить стан посівів та ґрунту.

На даний час автоматизовані теплиці тільки набирають популярності тому, що раніше готове рішення коштувало значні кошти, не всі могли собі дозволити придбати цю продукцію. З розвитком технологій і збільшенням доступності мікроконтролерів, на сучасному ринку з’явилися відносно дешеві пристрої для створення певних кліматичних умов.

1.2 Аналіз програмних продуктів – аналогів

Для того щоб сформулювати вимоги до нашого мобільного додатку, потрібно проаналізувати існуючи аналоги. Ми розглянимо чотири додатки: «2Agroclud», «Розумна теплиця №1», «Розумна теплиця №2», «Універсальний мобільний додаток до платформи Arduino». Серед аналогів наведено два різних додатки з однаковою назвою, тому ми пронумеровали їх.

1.2.1 Програмний додаток «2Agrocloud».

Програмний додаток [5] є частиною програмно-апаратного комплексу, який містить в собі веб-сайт компанії, обладнання та датчики, які можна придбати тільки в Інтернет-магазині компанії (рис. 1.2). Керування дозволяє управляти обладнанням теплиці по мережі.

Можливості:

– контроль і управління кліматом і CO_2;
– вимірювання дози освітленості і вмикання освітлення;
– контроль вологості ґрунту і зрошення.

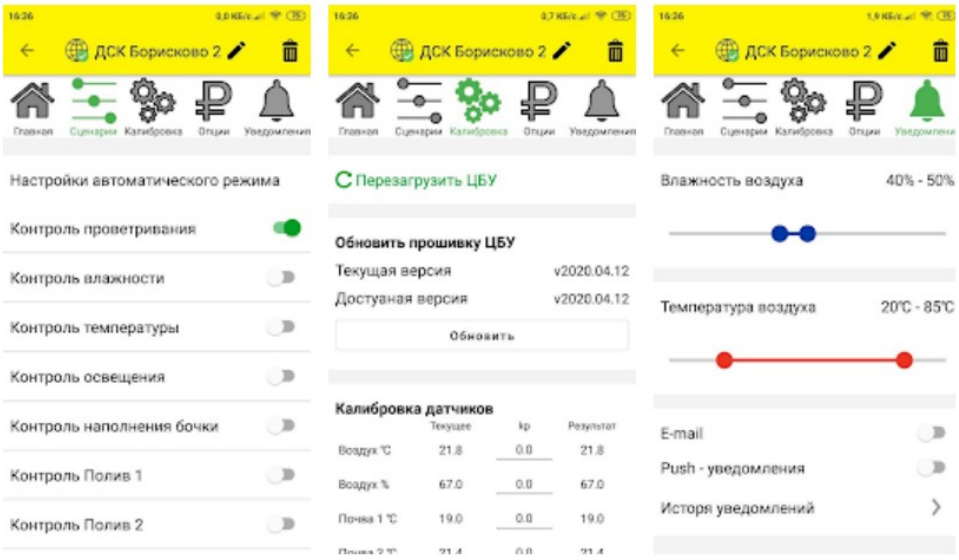


Рисунок 1.2 – Інтерфейс «2Agrocloud»

Способи зв'язку між контролером і мобільним пристроєм (додатком): Інтернет (Wi-Fi) з будь-якого місця через хмарний сервер.

1.2.2 Програмний додаток «Розумна теплиця №1».

Програмний додаток [6] дуже схожий з попереднім додатком, бо також зав’язаний на програмно-апаратному комплексі, і потребує купівлі спеціального обладнання (рис. 1.3). Програмне забезпечення автоматизованої системи керування дозволяє управляти обладнанням теплиці по мережі. Подіями для керування вузлами теплиці можуть служити команди користувача, розклад в календарі або ж дані з датчиків. Дозволяє масштабувати систему шляхом збільшення кількості підтримуваного устаткування.

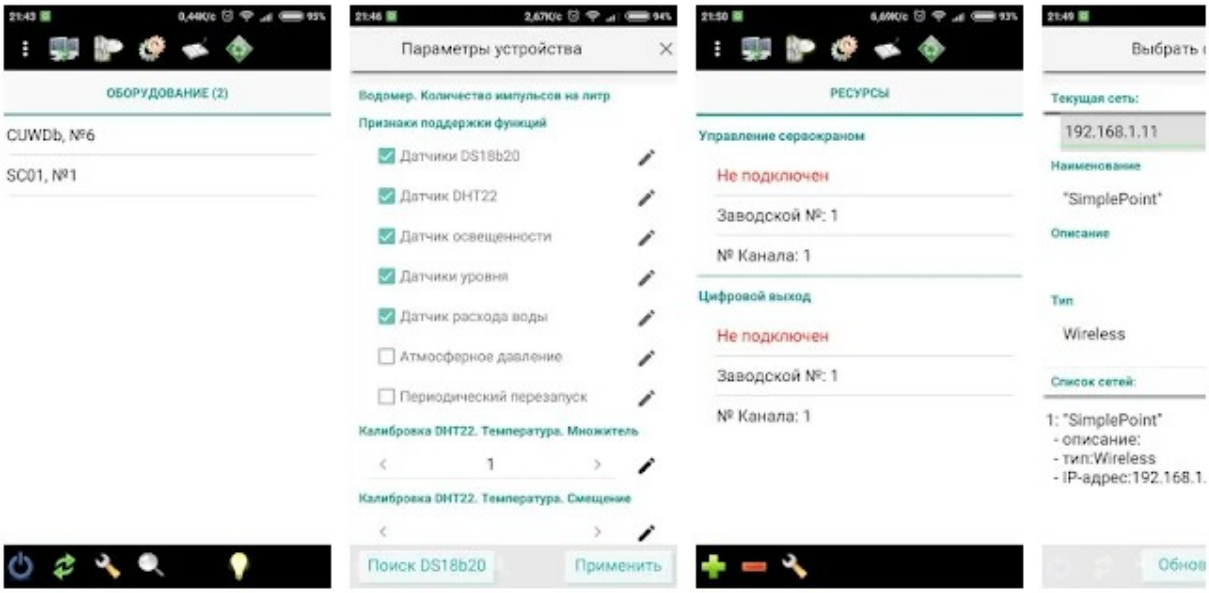


Рисунок 1.3 – Інтерфейс «Розумна теплиця №1»

Способи зв'язку побудовані через мережу Інтернет.

1.2.3 Програмний додаток «Розумна теплиця №2».

Програмний додаток [7] був створений для комунікацій з певним приладом, який можна придбати тільки у автора додатку. «Розумна теплиця» (рис. 1.4) дозволяє автоматично виконувати наступні функції: контроль і регулювання температури всередині теплиці, моніторинг вологості повітря, чотири незалежні зони поливу, додаткове освітлення при нестачі природного світла, вбудований журнал на 2000 подій.

Спосіб зв'язку між контролером і мобільним пристроєм: Bluetooth.

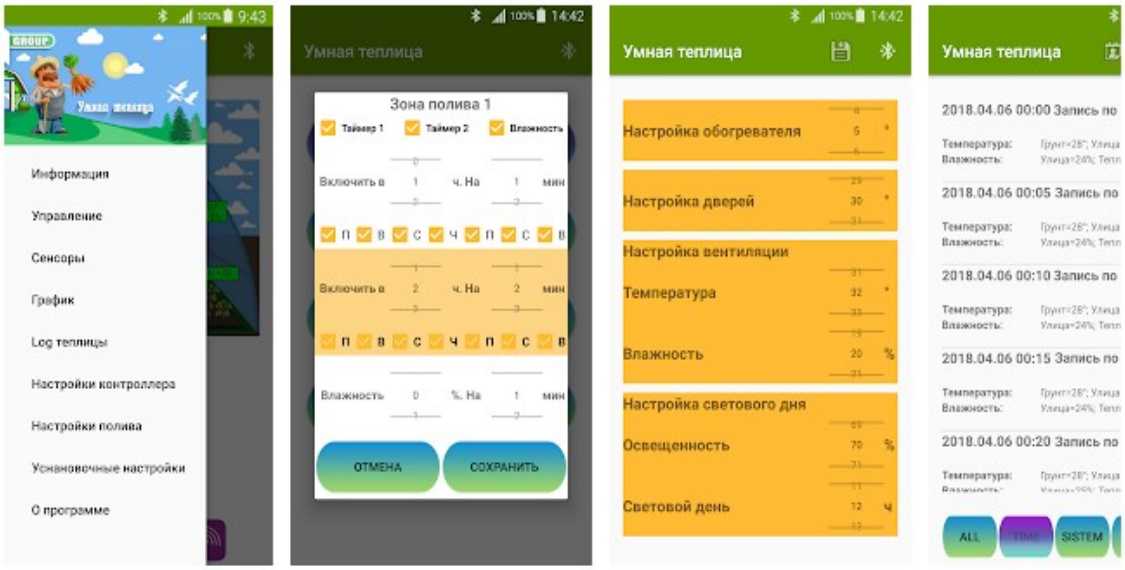


Рисунок 1.4 – Інтерфейс «Розумна теплиця»

1.2.4 Програмний додаток «Універсальний додаток до «Arduino».

Програмний додаток [8] сотворений для використання плати «Arduino», яка в свою чергу виконує центральну роль (рис. 1.5). За допомогою донного додатку (рис 1.6) отримуються данні від плати «Arduino»:

– температура і вологість повітря;
– температура і зволоженість ґрунту;

– освітленість теплиці.

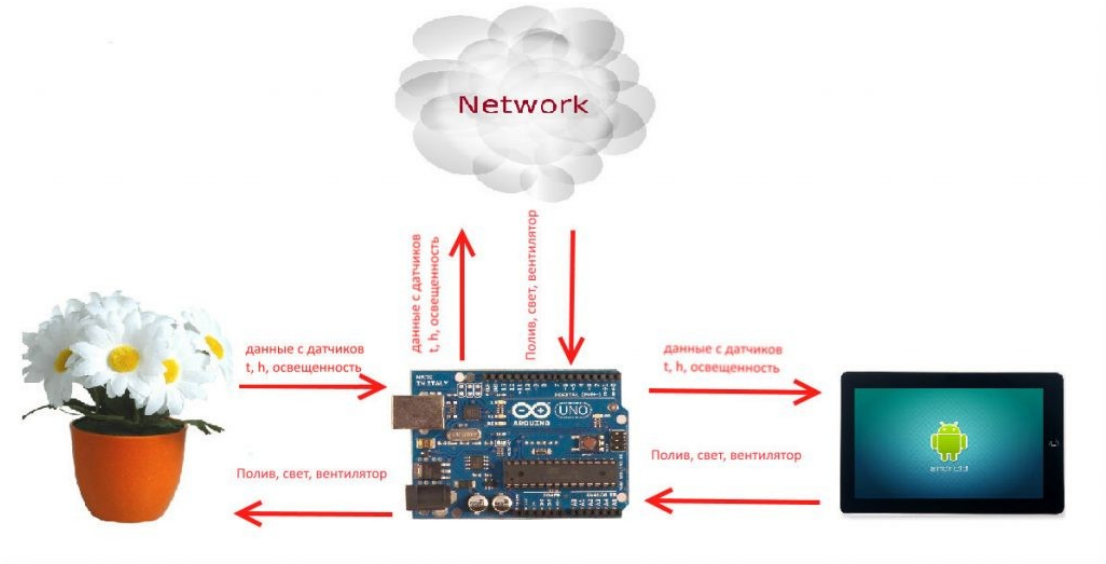


Рисунок 1.5 – Схема, яка побудована на контролері «Arduino» .

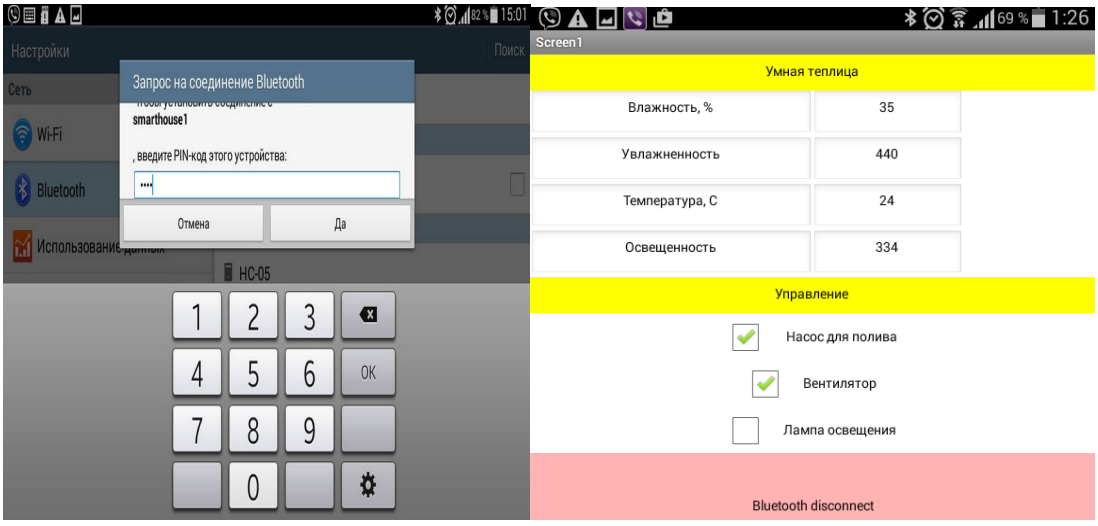


Рисунок 1.6 – Інтерфейс мобільного додатку «Arduino»

Способи зв'язку між контролером і мобільним пристроєм (програмним застосунком): Bluetooth.

Детально розглянувши дані програмні застосунки, можна виділити деякі особливості, які представлені в таблиці 1.1.

Таблиця 1.1 – Порівняльний аналіз аналогічних розробок

Назва додатку Критерій	2Agrocol	Розумна теплиця №1	Розумна теплиця №2	Універсальний додаток до Arduino	Майбутня розробка
Наявність авторизації (повна або часткова)	+	+	+	+	+
Наявність реєстрації	+	+	-	-	+
Особистий кабінет	+	-	-	-	-
хмарний сервер	+	+	-	-	+
Push-повідомлення	+	+	+	-	+
Потребує платній сервіс	+	-	-	-	-
Працює з будь-якою кількістю об’єктів	+	-	-	-	+
Фіксованій та обмеженій моделями набір виконавчих приладів	-	+	+	-	-
Фіксованій та обмеженій типами та кількістю набір параметрів для керування	-	+	+	-	-

Продовження таблиці 1.1

Назва додатку Критерій	2Agrocol	Розумна теплиця №1	Розумна теплиця №2	Універсальний додаток до Arduino	Майбутня розробка
Широкий набір параметрів налаштувань	+	+	+	-	+
Інтерфейс	+	+	+	-	+
Адаптивність до різноформатних екранів	+	+	+	-	+

Зазначені програмні застосунки, та аналогічні розробки, написані під конкретні апаратні можливості виконавчих механізмів та завдань агрокомпаній, фізичних осіб. Проаналізувавши представлені додатки ми можемо сформулювати функціональні вимоги до власного програмного продукту, а саме:

– вхід до облікового запису шляхом авторизації на початку роботи додатку;

– обрання потрібної теплиці, якщо їх декілька;

– перегляд списку характеристик та значень, параметрів, характеристик системи підтримки діяльності теплиці;

– перегляд архівних значень параметрів характеристик системи підтримки діяльності теплиці у вигляди графіку;

– редагування налаштувань керуючих параметрів системи підтримки діяльності теплиці шляхом вибору дискретних значень з фіксованого діапазону;

– вибір мови додатку.

1.3 Постановка задання

Метою проекту є створення мобільного додатку підтримки діяльності садової теплиці, що дозволить користувачу додатку спостерігати за показниками садової теплиці у реальному часі та збереженими в архів даними, вносити зміни в керуючий прилад. При запуску мобільного додатку користувач матиме можливість продивитися показники, а також обрати показник. Головним завданням для досягнення мети є:

- розвернутий аналіз предметної області та аналогічних додатків для виявлення переваг та недоліків, формулювання функціональних вимог;
- розгляд існуючих технологій розробки, вибір на користь технології розробки, у якої більше переваг;
- проектування: реалізація моделі варіантів використання із визначенням акторів системи та ВВ., структурно-функціональне моделювання в нотації IDEF0 для моделювання бізнес-процесів, які протікають в додатку;
- розробка мобільного додатку, впровадження, тестування на декількох пристроях.

В результаті аналізу вхідних даних та вимог був сформований документ специфікації вимог до програмного продукту, який наведений в додатку А.

1.4 Вибір засобів реалізації

1.4.1 Методи реалізації мобільного додатку

Основна робота прикладної програми полягає в завантаження даних зі сховища даних, оновлення інтерфейсу користувача та додавання нової

інформації на основі запитів користувача. Тому, має сенс спів підставити компоненти інтерфейсу користувача з компонентами сховища даних. Метою такої схеми є поділ компонентів на: компоненти інтерфейсу користувача (View), компоненти, які забезпечують основні функціональні можливості (Controller), та дані (Model). Можна зробити висновок, що на рівні стандартної мобільної прикладної програми модель MVC [9] добре вписується в Android та IOS системи. Традиційний підхід до реалізації проектів у вигляді каскадної моделі, яка передбачає поетапне просування до мети, не задовольняє нашим умовам розробки програмного продукту: малий обсяг проекту, обмеження у часі розробки. Вимоги зрозумілі, але не стабільні та можуть доповнюватись. Тому потрібно використовувати більш гнучкі моделі проектування додатку. Недоліком гнучких моделей є незрозуміла структурованість підходів. Орієнтованість нашого програмного продукту на клієнта, тому що вона передбачає його безпосередню участь в процесі роботи, доцільно застосувати методику Scrum [10]. Scrum робота ведеться короткими циклами, застосовувати цю методику, можливо на самому початковому етапі. Що дозволяє усунувши помилки, підтримувати постійний зв'язок з замовником. Що виключає створення непотрібного йому товару. Враховуючі мало численність команди розробників (одна людина), можливості у розщепленні роботи на окремі потоки немає. Тому обирається *rapid application development* (RAD) [11] модель проектування. Це надає швидку реалізацію частки функцій системи у вигляді прототипу, його перевірку та використання у наступному прототипі, контроль за ризиками на ранніх етапах розробки тощо. Недоліки RAD – це слабкий дизайн на ранніх етапах розробки.

Враховуючи, що необхідно в процесі розробки керувати завданнями окремого розробника і, в майбутньому, команди в цілому, показувати, які артефакти необхідно розробити, аналізувати критерії відстеження та вимірювання продуктів і функціонування проекту, логічно було б

застосувати уніфікований процес розроблення ПЗ *Rational Unified Process* (RUP[12]). Проект системи буде складатися у сукупності моделей, складених за CASE – технологією, занотованих в UML.

1.4.2 Обрання засобів реалізації

Мобільний додаток, який створюється має бути уніфікований для усіх гаджетів з операційною системою Android та IOS, які є топами у світі на даний час. Створення подібного продукту тягне за собою дослідження і пізнання в таких предметних областях, як мережеві протоколи передачі даних, програмування для мобільних пристроїв на мові програмування, розробка дизайну для мобільних додатків, проектування інтерфейсів мобільних додатків. Створення мобільних додатків для Android на об’єктно-орієнтованій мові, найчастіше виконуються на Java , на якій розроблено 90% та на інших. Серед інших набуває популярності нова мова Kotlin. Що до iOS платформи, то тут використовуються дві ведучі мови – перша Objective C, друга Swift. Отже, нам потрібно створити мобільний додаток, яке працює більше ніж на одній операційній системі або апаратній платформі, та є багатоплатформеним. Застосовуючи крос-платформну розробку мобільного додатку, отримуємо можливість получити більш швидкої розробки. Це достягається завдяки створення додатку з мінімальні знання нативного середовища створюючих мобільних додатків IOS та Android. Крос-платформне розроблення передбачає застосування спеціальних утиліт (фреймворків) для написання програми на основі сімейства мов JavaScript. Структура і логіка мобільного додатка створюється за допомогою інструментів :як PhoneGap, Titanium, Xamarin, Cordova на JavaScript. Потім це обертається в нативний виконуючий елемент, який інтегрується у базовий проект. Це дозволяє генерувати збірки проекту з однією і тією ж логікою під

декілька операційних систем відразу. Нам потрібен фреймворк, який розрахований для створення саме клієнтських програм, бажано SPA-рішень (Single Page Application). Тобто невеликих додатків з використанням сімейства мов JavaScript. Огляд сучасних ІТ-рішень вказує на Ionic framework – один із найширше обговорюваних фреймворків. Як повідомляє офіційний сайт [13], Ionic – це SDK для написання гібридних мобільних додатків, з набором CSS і JS компонентів, які створені на AngularJS, SASS, Apache Cordova. Мобільні додатки, створені за допомогою Ionic framework написані мовою TypeScript. Розроблений мобільний додаток системи підтримки діяльності теплиці генерує формалізований запит, надсилає його на сервер. Серверне програмне забезпечення обробляє цей запит і формує відповідь мобільному додатку. Це застосування клієнт-серверної архітектури передачі даних, тому кращим з простих протоколів передачі для цього мобільного додатку є протокол HTTP. До того ж протокол HTTP виконує роль транспорту для передачі самих даних у форматі JSON.

2 ПРОЕКТУВАННЯ МОБІЛЬНОГО ДОДАТКУ

2.1 Структурно-функціональне моделювання

Для упорядкування інформації ,що до проектування функцій роботи мобільного додатку було проведено моделювання роботи додатку.

Для опис функціональних-процесів, скористаймося методологією IDEF. Основним компонентом даної моделі є діаграми. Стандартом IDEF передбачається концепція побудови функціональних блоків та зв’язків між ними. Здійснюються моделювання контекстної діаграми, що представляє собою єдиний блок опису системи та її зв’язків з оточуючим середовищем, як передбачає методологією IDEF0 [14]

На рисунку 2.1 представлена контекстна діаграма в нотації IDEF0 мобільного додатку підтримки діяльності садової теплиці.

Рисунок 2.1 – Контекстна діаграма в нотації IDEF0

Наступний рівень IDEF1 діаграми має блоки та дуги. Блоки показують функції які повинна мати майбутня система, дуги в свою чергу вказують на взаємозв’язки між блоками, їх взаємодію. [15]

Системи на вході має запити до додатку на перегляд інформації, що до стану садової теплиці. Керування виконується шляхом запитів додатку на редагування налаштувань керуючих параметрів системи. Результатами роботи є інформування користувача про стан садової теплиці та можливості редагування керуючих параметрів.

Побудуємо діаграму в нотації IDEF1, яку декомпозуємо на підсистеми. Діаграма декомпозиція процесу роботу садової теплиці зображена на рисунку 2.2.

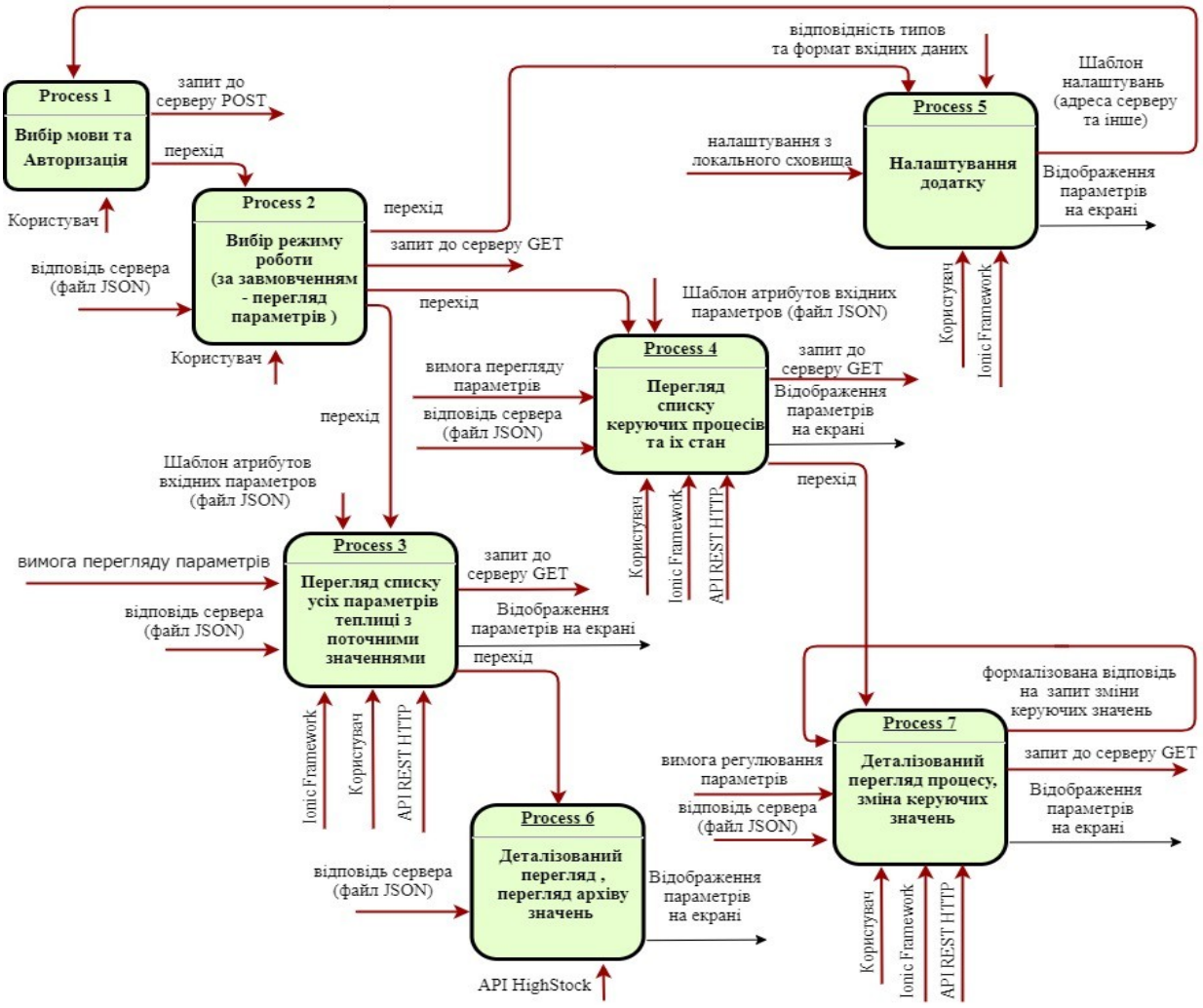


Рисунок 2.2 – Декомпозиція IDEF1-діаграми мобільного додатку

Діаграма IDEF1 декомпозована на наступні блоки:

– авторизацію та вибір мови додатку;

– вибір режиму роботи;

–перегляд списку усіх параметрів теплиці (завантажуються за замовчуванням);

– перегляд списку керуючих процесів та їх стан;

– налаштування додатку;

– деталізований перегляд та відображення архіву значень;

– деталізований перегляд обраного процесу та зміна керуючих значень.

Для функціонування блоків «авторизацію та вибір мови додатку» та «вибір режиму роботи» необхідно користувачу вести облікові данні користувача, обрати мову та після успішного входу обрати режим.

Для функціонування блоку «перегляд списку усіх параметрів теплиці» мобільний додаток попередньо формалізований запиту на перегляд інформації. Виходом є інформація про список усіх параметрів садової теплиці та їх поточних значень.

Для виконання блоку «перегляд списку керуючих процесів та їх стан», необхідна виконання запиту на перегляд інформації. На виході з цього блоку ми отримуємо інформацію про їх стан. Підсумком роботи блоку є відображення списку керуючих процесів.

Для виконання роботи блоку «деталізований перегляд обраного процесу та зміна керуючих значень» мобільний додаток попередньо формалізований запиту на перегляд інформації. Виходом є інформація про налаштувань керуючих параметрів.

2.2 Моделювання варіантів використання мобільного додатку

Окремим результатом моделювання додатку буде діаграма варіантів використання (Use Case), яка демонструє звязок між акторами системи та множиною сценаріями поведінки.

Для того щоб побудувати діаграму варіантів використання опишемо модель варіантів використання.

Опис акторів:

– користувач – авторизована особа, яка використовує програму щодо перегляду характеристик теплиці, введення нових обмежень параметрів, редагує існуючі настройки мобільного програмного застосунку;

– гість – неавторизована особа, яка переглядає інструкцію користувача;

– веб-ресурс – віддалена ресурс програма, яка забезпечує обробку запитів та надання даних щодо параметрів і характеристик системи підтримки діяльності теплиці;

– highcharts – API для побудови графіків (плагін);

– сховище – локальне сховище для збереження налаштувань.

Знаючи всіх акторів мобільного додатку садової теплиці перейдемо до реалізації функціоналу мобільного додатку.

Для реалізації функціоналу програми визначимо варіанти використання за допомогою таблиці 2.1.

Таблиця 2.1 – Варіанти використання

Варіанти використання	Актор, який приймає участь у ВВ				
	Користувач	Веб-ресурс	Highcharts	Гість	Сховище
UC01 Log In (Авторизація)	×	×			×
UC02 LoadParam (Показ та вибір характеристик)	×	×			
UC03 LoadValue (Показ поточних та архівних значень параметрів)	×	×	×		
UC04 EditValue (Регулювання значення параметрів)	×	×			
UC05 EditSetting (встановлення налаштувань)	×			×	×
UC06 About (Дані о програмі)	×			×	
UC07 Exit (Вихід з програмного застосунку)		×		×	

Примітка : UC02, UC03 або UC04 може бути виконаний тільки, якщо виконаний UC01.

Аналіз вимог реалізиції варіантів використання дозволяє запропонувати наступну діграму використання мобільного додаткупідтримки діяльности тепліци (рис. 2.3).

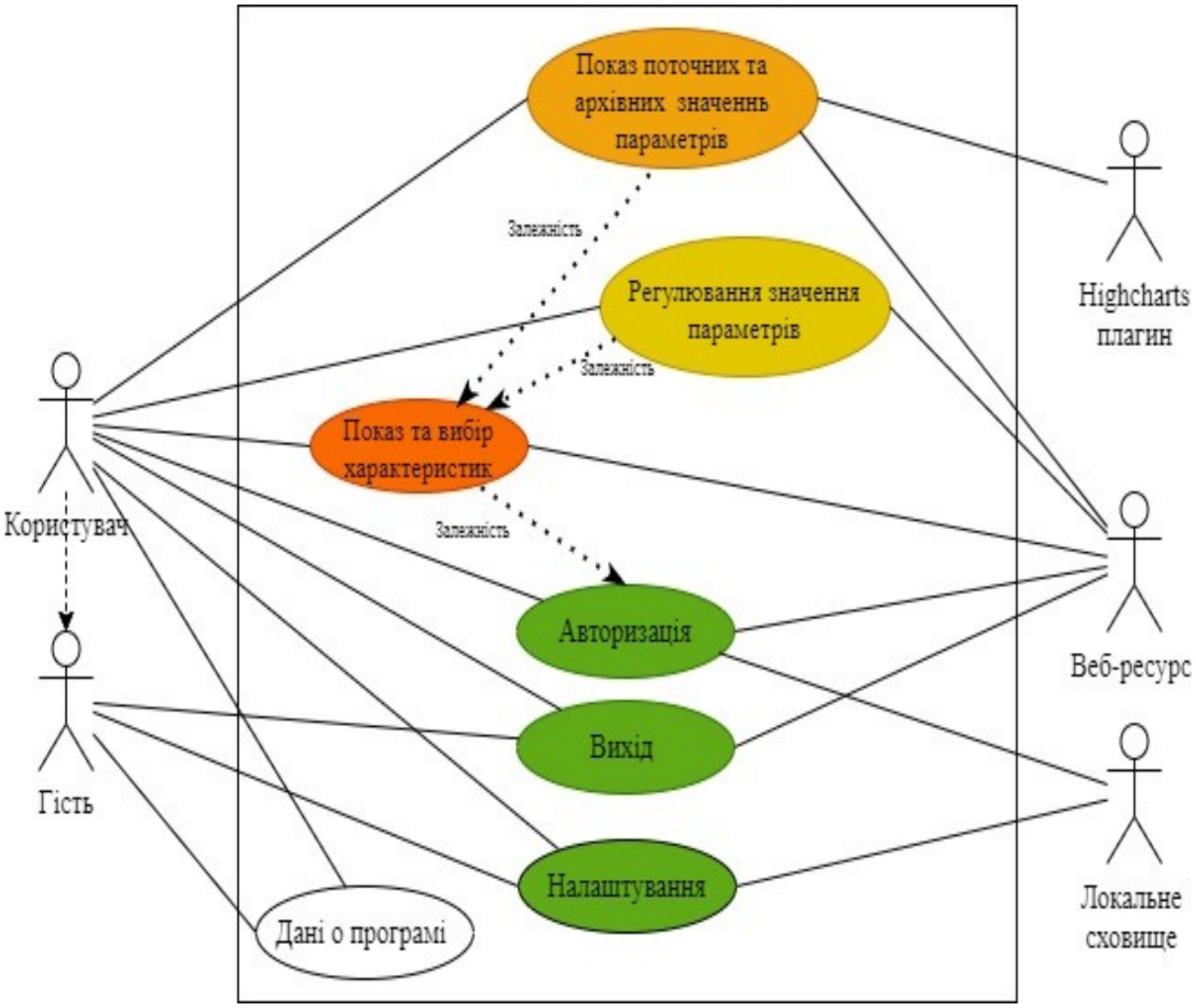


Рисунок 2.3 – Діаграма сценарію використання мобільного додаткупідтримки діяльности тепліци

2.3 Модель аналізу мобільного додатку садової теплиці

Так як мобільний додаток є частиною інформаційної системи підтримки діяльності теплиці(яка вданий час не розглядається), то статичних представлень абсолютно недостатньо для моделювання процесів функціонування подібних систем як в цілому, так і їх окремих підсистем і елементів. Для специфікації функціональності варіантів використання, нам

необхідно описати можливі послідовності станів і переходів, які в сукупності характеризують поведінку моделі мобільного додатку протягом його життєвого циклу. Діаграма станів (рис. 2.4 - 2.5) представляє динамічну поведінку сутностей, на основі специфікації їх реакції на сприйняття деяких конкретних подій.

Рисунок 2.4 – Діаграма станів мобільного додаткупідтримки діяльности тепліци при успішної авторизації

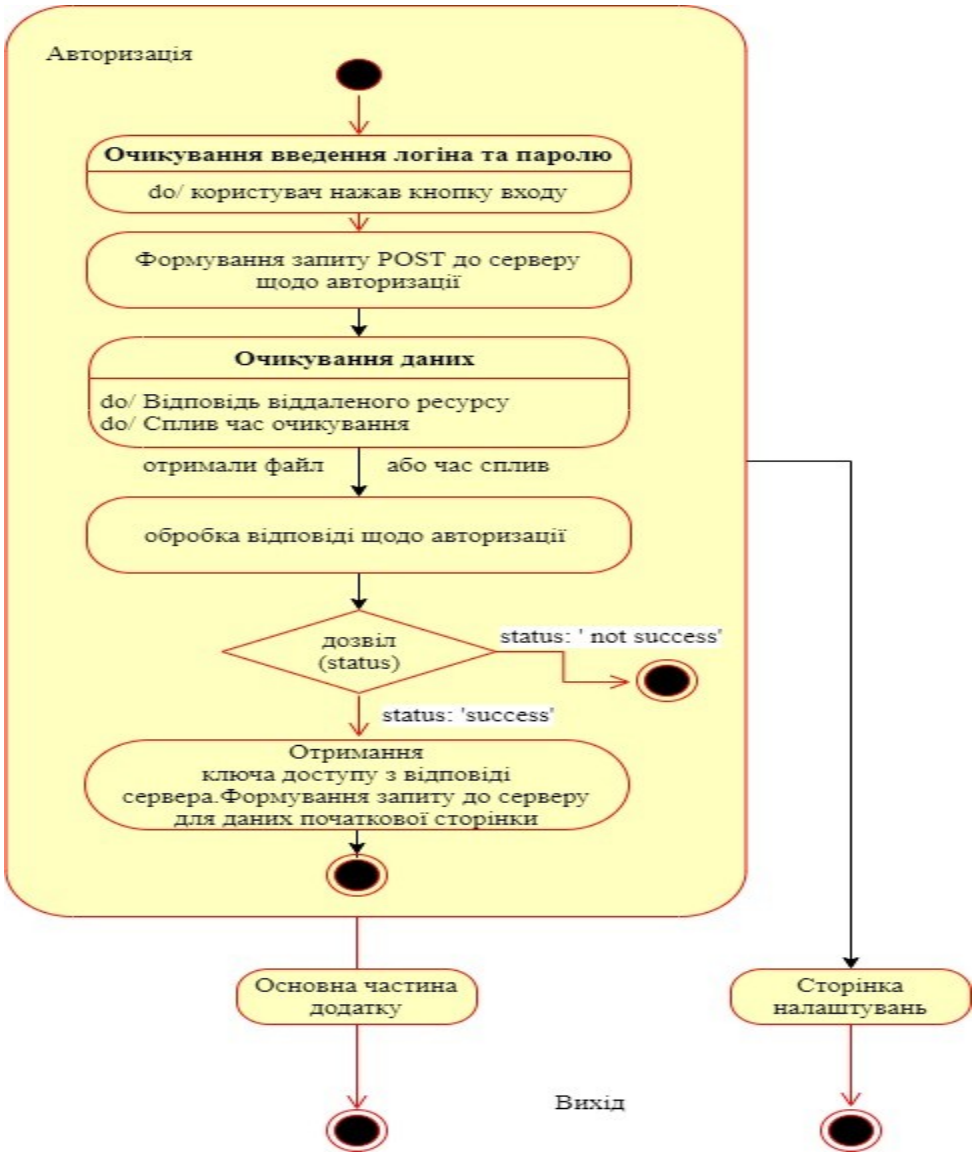


Рисунок 2.5 – Діаграма станів мобільного додаткупідтримки діяльности тепліци при авторизації

Провівши аналіз декомпозиції контекстної діаграми IDEF0, варіантів використання та діаграм станів, з метою створення коду мобільного додатку, застосовуючи методи об'єктно-орієнтованого програмування, необхідно визначити послідовність дій та класів об’єктів. Побудуємо способом формалізації сценаріїв діаграми послідовності, в якої опишемо послідовність дій, ініціатором яких є користувачем(Рис. 2.6). У подальшому компоненти і

потоки повідомлень будуть трансформовані конкретні класи (об’єкти) та методи цих об’єктів.

.

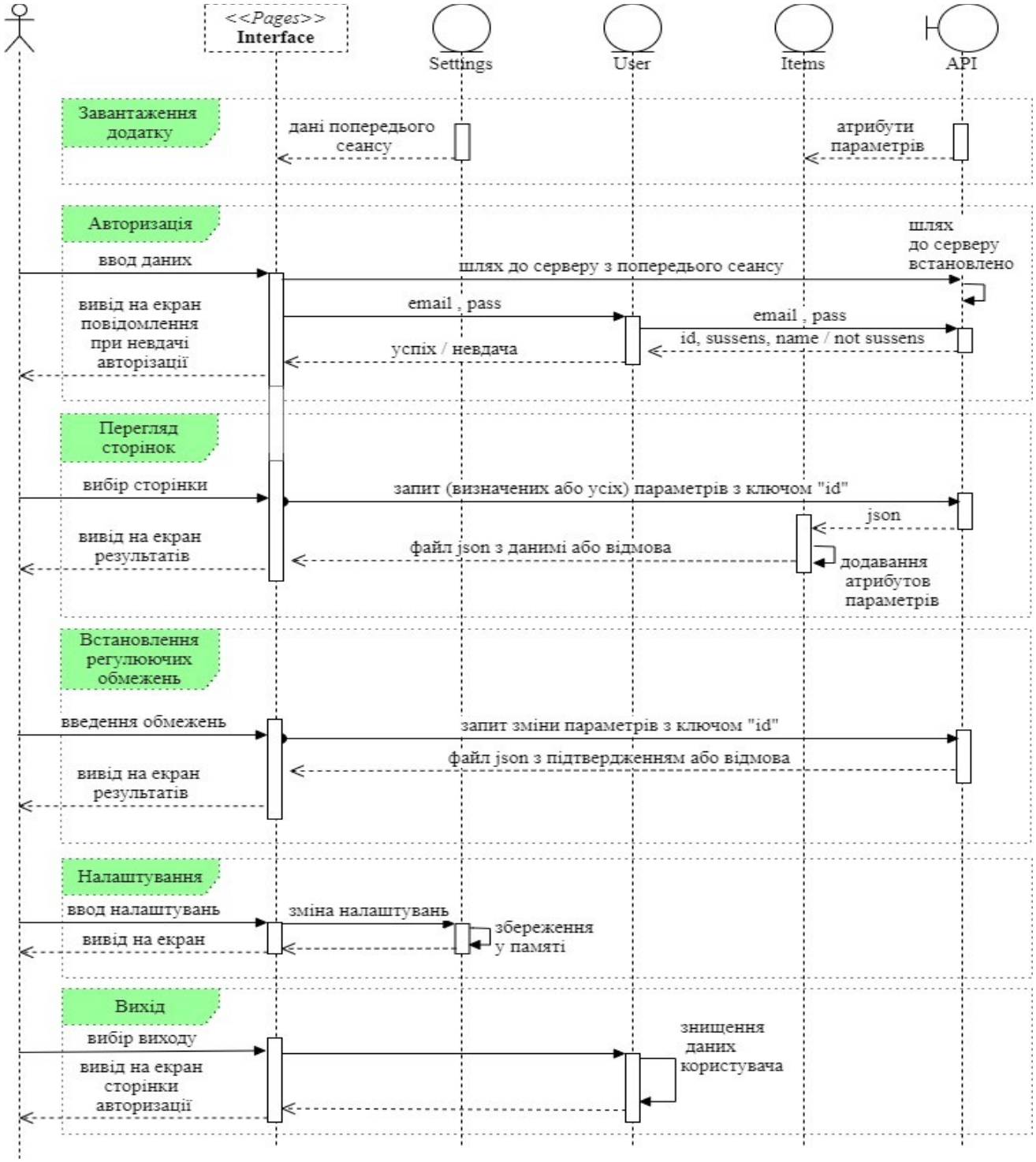


Рисунок 2.6 – Діаграма послідовності мобільного додаткупідтримки діяльности тепліци

3 РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ ПІДТРИМКИ ДІЯЛЬНОСТІ САДОВОЇ ТЕПЛИЦІ

3.1 Архітектура мобільного додатку

До мобільного додатку застосуємо модульну архітектуру.

Поділімо роботу всередині коду програми на шари - програмні модулі, які розміщені в різних папках або каталогах на стороні клієнта.

Трьохшарова архітектура включає в себе:

Presentation Layer.- шар, код якого найбільш тісно взаємодіє з користувачем. Це шар уявлення (графічний інтерфейс) побудуємо на основі шаблону проектування MVC[16], якій зображено на рисунку 3.1.

Domain Layer- .шар який опрацьовує робочу логіку коду програмного застосунка.

Data Layer. шар доступу до даних (зв'язок з базою даних або джерелом даних) побудуємо на основі REST компонента Ionic API (рис. 3.2).

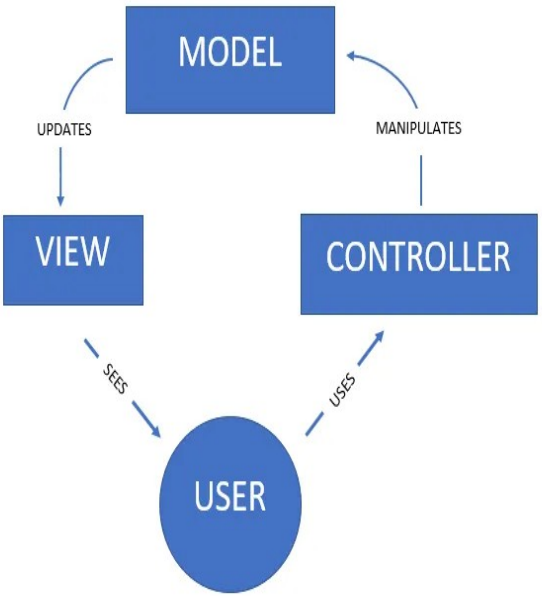


Рисунок 3.1. – Архитектура MVC

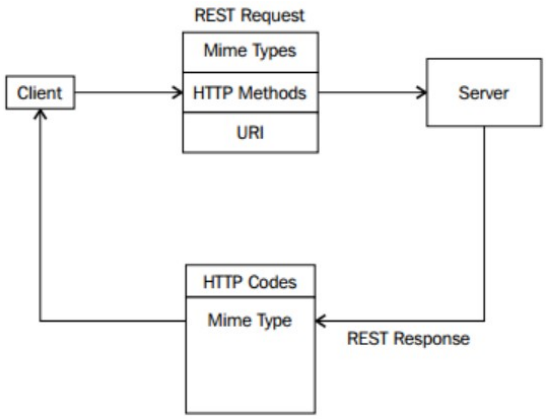


Рисунок 3.2. – Приклад реалізації REST компонента у програмному застосунку

Більшість приватних малих домогосподарств, які утримують сади або теплиці мають системи автоматизації з малої кількістю параметрів, які контролюються (5-10 параметрів). Кількість параметрів, а відповідно і датчиків, заздалегідь відома вже під час визначення посадкового сільськогосподарського матеріалу. Крім того більшість приладів контролю та виконавчих приладів мають примітивні протоколи передачі даних та не мають можливості самостійного з’єднання з Інтернет ресурсами. Контролери, які керують системами автоматизації, як правило мають власний спосіб додавання датчиків, виконавчих приладів та користувачів. Тому прийняте рішення не закладати зайві функції в програмний застосунок, що розроблюється, а обмежитись вичерпним переліком.

Застосовуючи наведену архітектуру мобільного додатку досягнемо ефекту спрощення користування по відношенню к аналогічними розробками : «2Agrocloud», «Розумна теплиця» №1 та №2. Одночасно ми будемо мати більш зручний користувацький інтерфейс по відношенню до мобільного додатку «Моніторинг і управління теплицею з телефону або планшета З ОС ANDROID».

3.2 Реалізація мобільного додатку

Створення мобільного додатку відображення діяльності садової теплиці складається з декількох етапів: створення функціональних модулів системи та створення сторінок мобільного додатку.

Першим кроком було створення функціональних модулів системи.

За допомогою діаграми класів (рис. 3.3) будується структура системи, описується спадкування й взаємне положення класів друг щодо друга. Тут зазначається логічне представлення системи, тому що класи потім будуть визначені як фізичні об'єкти, з функціями і методами.

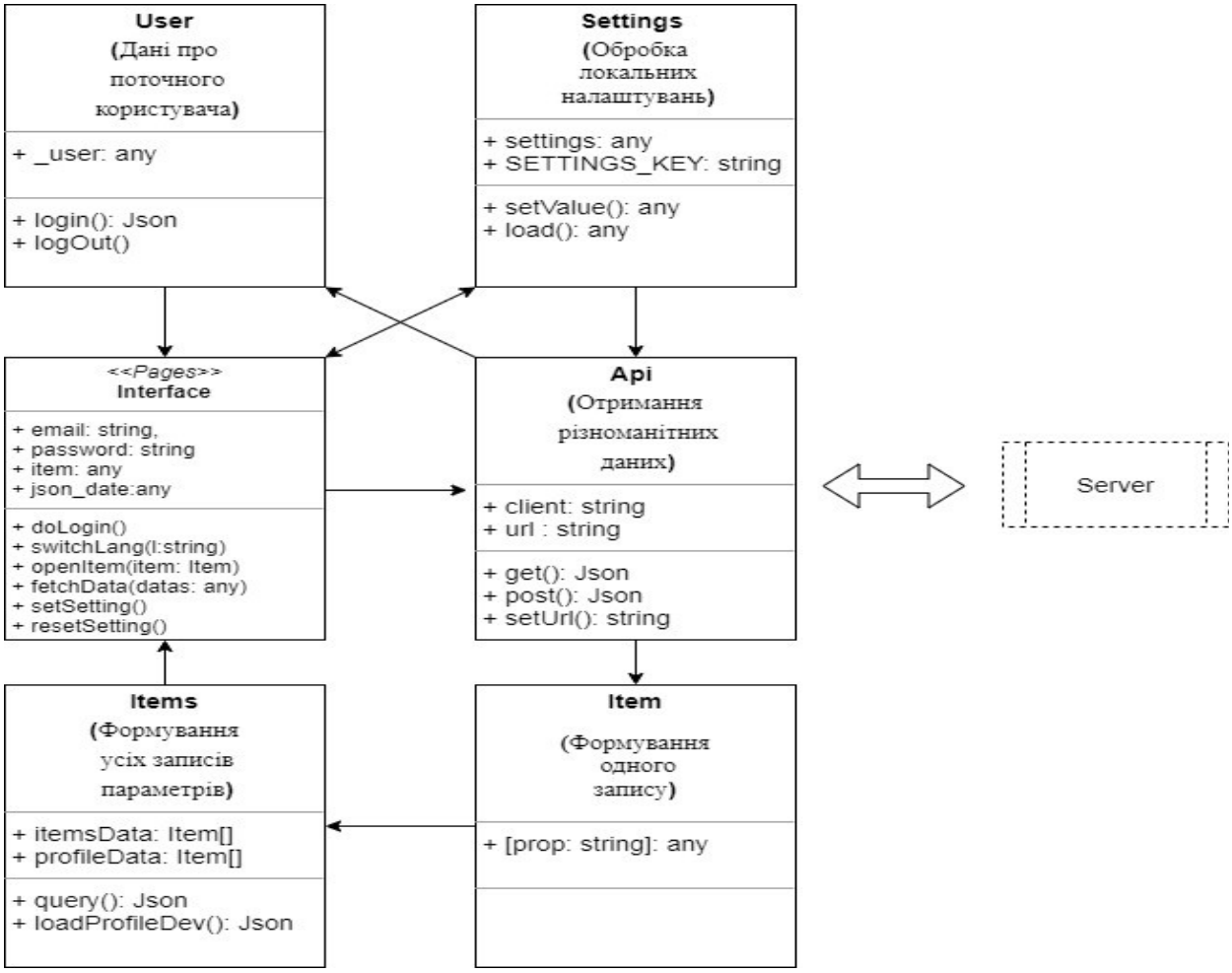


Рисунок 3.3 – Діаграма класів мобільного додатку підтримки діяльности тепліци

Наступним кроком було створено дизайн сторінок. Визначивши технологію, через яку будемо реалізації мобільний додаток, об’єкти додатку та предметний напрямок можемо розпочати прототипування сторінок мобільного додатку, які входять до класу interface - *Pages*.

За допомогою сервісу «AppMaster.io»[17] було спроектовано основні екранів мобільного додатку з метою визначення кращого розташування заголовків сторінок та кнопок навігації. Інтерфейс платформи «AppMaster.io» показано на рисунку 3.4.

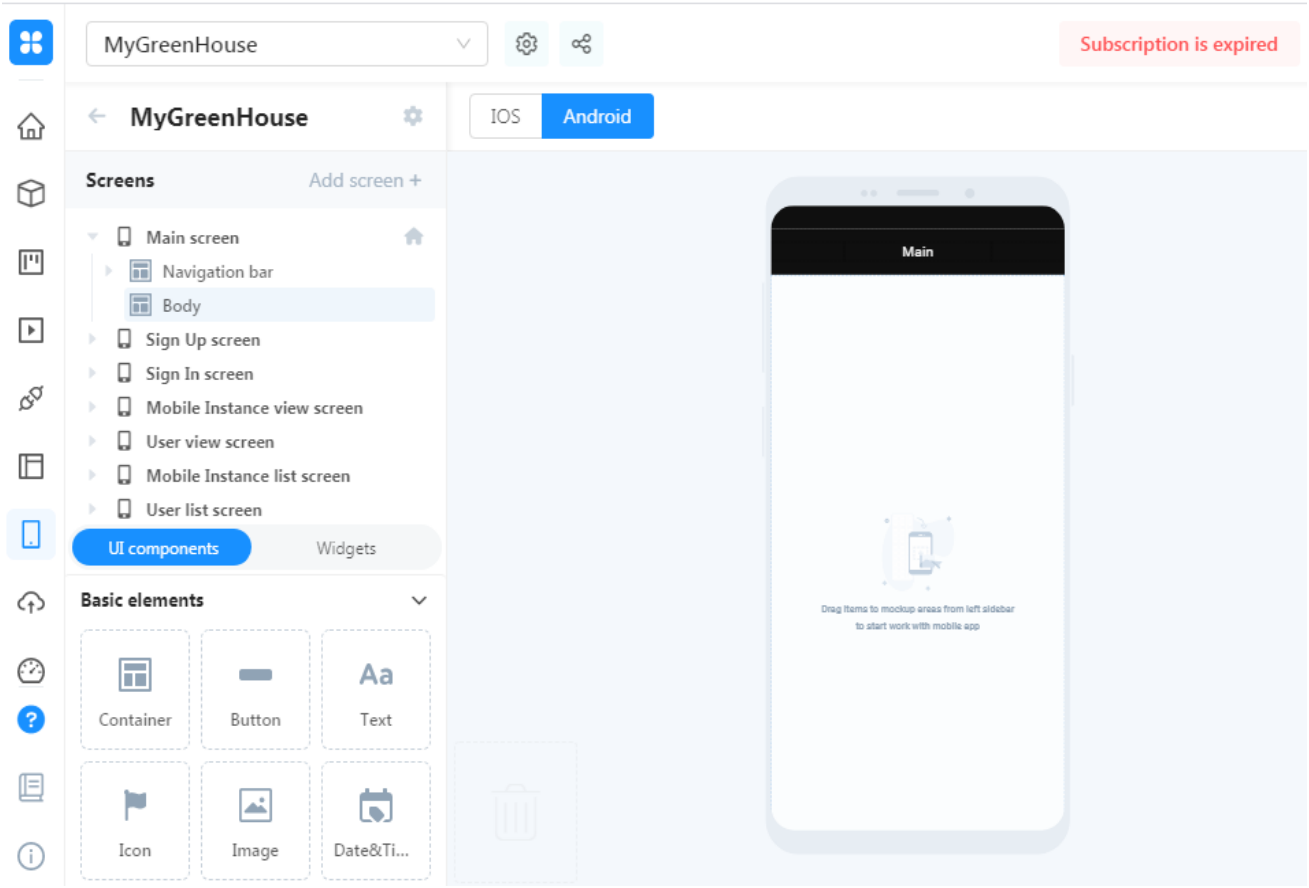
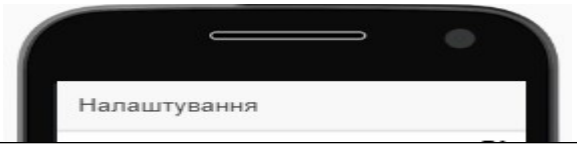
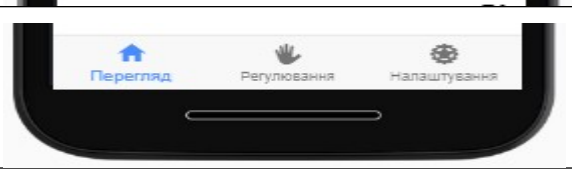


Рисунок 3.4 – Інтерфейс для прототипування мобільних програмних застосунків в середині платформи «AppMaster.io»

Таблиця 3.1 – Макети користувацького інтерфейсу

Макет№1 Заголовок сторінки «Перегляд»	
Макет№2 Заголовок сторінки «Регулювання»	
Макет№3 Заголовок сторінки «Налаштування»	
Макет№4 Компонент перемикача режимів роботи	

Даний проект був створений з використанням Ionic.v3 [18]. При створенні мобільного додатку на Ionic створюється структура проекту (рис. 3.5) в якої файли зі шрифтами, файли перекладів, зображення, провайдери, сторінки та компоненти знаходяться в окремих теках. Файли знаходяться в теці src (рис. 3.6).

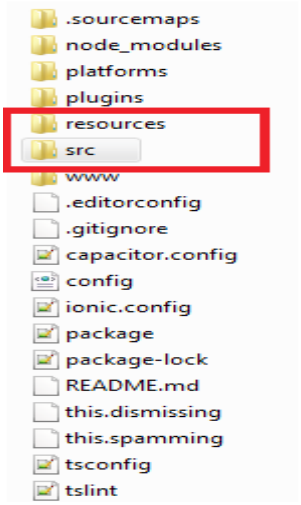


Рисунок 3.5 – Розташування файлів проектів Ionic.v3

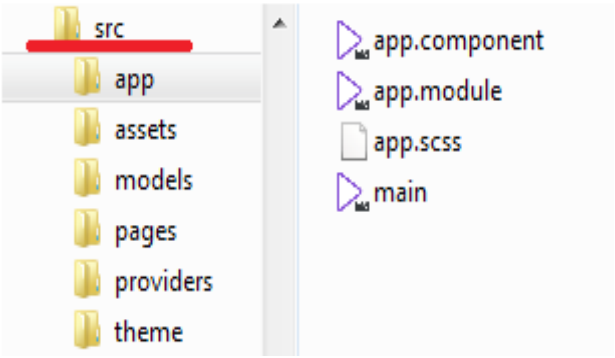


Рисунок 3.6 – Тека src

У текі src знаходяться файли модуля app, має у собі підключення всіх інших файлів та компонентів мобільного додатку. При запуску мобільного додатку першим запускається app.component.

У теці resources (рис. 3.7) зібрані файли зображень, що будуть застосовані підчас запуску мобільного додатку для сплеш екрану та інше.

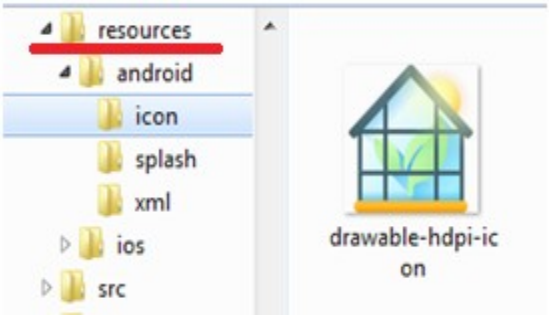


Рисунок 3.7 – Тека resources

Реалізуючи діаграму класів для написання коду функцій необхідно створити файли які будуть групувати в собі функціонал логіки програмного застосунку. Їх називають провайдери. Особливістю провайдерів є те, що вони ініціалізуються один раз і залишаються в пам’яті до завершення роботи мобільного додатку, тому це один і не дублюється.

Для проекту необхідно створити 4-х провайдерів (рис. 3.8): Api, Items, Settings, User.

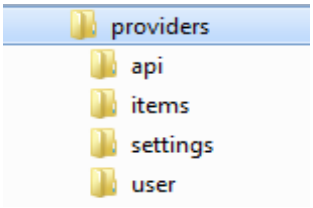


Рисунок 3.8 – перелік провайдерів

Для розкриття функції кожного провайдери запишємо їх у таблиці 3.2.

Таблиця 3.2 – функцій провайдера

Провайдер	функції	Опис функціоналу
Api	post()	Функція відправлення даних о користувачеві та отримання його ідентифікатору
	get()	Функція відправлення запитів або даних та отримання відповіді серверу
	setUrl()	Функція встановлення шляху до серверу
Items	query()	Функція генерує загальні запити даних щодо пристроїв або процесів з серверу, та формує список для виводу на екран
	loadProfileDev()	Функція завантаження профілів,а саме ТТХ пристроїв (датчиків), з зовнішнього файлу
Settings	load()	Функція відкриття та підключення змінної локального сховища
	allSettings()	Функція завантаження усіх значень змінної локального сховища одночасно
	setValue()	Функція встановлення та запису нового значення змінної локального сховища
User	Login()	Функція перевірки даних о користувачеві, отримання підтвердження авторизації та кодів доступу до сеансу обміну даними
	logout()	Функція знищення даних о користувачеві та кодів доступу сеансу обміну даними

Для реалізації класу інтерфейсу користувача необхідно створити сторінки нашого мобільного додатку(рис. 3.9).

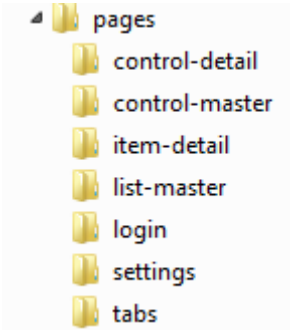


Рисунок 3.9 – Сторінки застосунку

Однією з особливостей ionic проектів є сутність побудови сторінки. Навігацію по мобільному додатку здійснюється лише по сторінкам. Однак «сторінки» це не один файл, а декілька які збираються при відображені. Використовуючи автоматичну генерації фрейндверку ionic при створені сторінки отримуємо 4 файли: модуля сторінки(page.module.ts), основний класу сторінки(page.ts), інтерфейс сторінки(page.scss), стилі сторінки(page.html).

Розглянимо детально структуру сторінок на прикладі сторінки list-master (рис. 3.10).

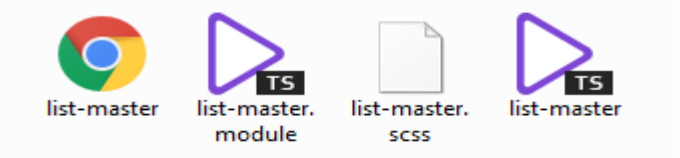


Рисунок 3.10 – Структура сторінки list-master

Далі розглянимо модулів та формат основний класу сторінки, що зображена на рисунку 3.11. Позначкою 1 позначено залежності сторінки модулів і провайдерів ,від інших модулів і провайдерів. Позначкою 2 визначена файл інтерфейсу якої буде підключатися до цієї сторінка. Позначкою 3 визначена ініціалізація системних та користувацьких провайдерів.

```
import { Component } from '@angular/core';
import { IonicPage, NavController } from 'ionic-angular';

import { Item } from '../../models/item';
import { Items } from '../../providers';
import { LoadingController } from 'ionic-angular';

@IonicPage()
@Component({
  selector: 'page-list-master',
  templateUrl: 'list-master.html'
})

export class ListMasterPage {
  currentItems:Item[]= [];

  constructor(public navCtrl: NavController, public items: Items,
              public loadingCtrl: LoadingController) {}

  //функція запускається, коли сторінка повністю введена і тепер є активною сторінкою.
  ionViewDidEnter(){
      const my_url = 'index.php?dev=0'
      this.currentItems = this.items.query(my_url);     // завантаження усіх даних
      }

  /**
   * функція для переходу на сторінку з детальною інформацією про вибраний елемент.
   */
  openItem(item: Item) {
    this.navCtrl.push('ItemDetailPage', {
      item: item
    });
    console.log('openItem ready ');
  }
}
```

1

2

3

Рисунок 3.11 – Структура файлу класу сторінки (list-master.ts)

Розуміючи вигляд майбутнього мобільного додатку, оперуючись на структуру проекту на Ionic Framework, розпочато створення розмітки всіх сторінок мобільного додатку використовуючи HTML5 та CSS3. Розміщення інформації та елементів керування на екрані мобільного пристрою здійснюється за допомогою файлу html, в якому визначена розмітка кожної сторінки. Зміст файлу html сторінки list-master зображено на рисунку 3.12.

Цифрою 1 позначено теги, якій використовується для виведення даних з серверу у вигляді послідовних записів.

Для функціонування мобільного додатку було розроблено 6 основних сторінок, 1 бічна сторінка та 1 елемент навігації:

- list-master. Сторінка створена для відображення списку параметрів, які є в системі підтримки діяльності теплиці для перегляду та вибору, з наступним детальним переглядом архівних значень;
- item-detail. Сторінка створена для відображення значень обраного параметру з детальним переглядом архівних значень у вигляді графіку;
- control-master. Сторінка створена для відображення списку процесів, які є в системі підтримки діяльності теплиці для перегляду та вибору, з наступним детальним переглядом та регулювання обмежувальних значень;
- control-detail. Сторінка створена для відображення статусу процесу та введення обмежувальних значень;
- login. Сторінка відображення текстових полів для авторизації з можливим вибором мови спілкування та наступним переходом до роботи в мобільному додатку;
- settings. Сторінка створена для відображення списку налаштувань мобільного додатку, зміна та введення нових значень налаштувань;
- tabs. Елемент сторінки, якій накладається на інші та використовується для навігації в мобільному додатку (реалізація макету № 4 таблиці 3.1). Він завантажується після авторизації у файлі сторінки login.ts . Інші макети №№1,2,3 реалізуються тегами <ion-header> на кожної сторінці, наприклад рисунку 3.12.


```
<ion-header>
  <ion-navbar>
    <ion-title>{{ 'LIST_MASTER_TITLE' | translate }}</ion-title>
  </ion-navbar>
</ion-header>
<ion-content class="ion-padding">
   <ion-list>
    <ion-item-sliding *ngFor="let item of currentItems">
      <button ion-item (click)="openItem(item)">
    <ion-row>
       <ion-col col-2>
        <ion-avatar>
          <img [src]="item.profilePic" />
        </ion-avatar>
       </ion-col>
       <ion-col col-6>
            <h2>{{item.name | translate}}</h2>
       </ion-col>
       <ion-col>
        <h2 >{{item.value}}{{item.suffix}}</h2>
       </ion-col>
    </ion-row>
      </button>
    </ion-item-sliding>
  </ion-list>
</ion-content>
```

Рисунок 3.12– Зміст файлу html сторінки list-master (list-master.html)

Бічна сторінка реалізована в файлі app.component.ts (рис 3.13, цифра 1), як така що є глобальною для мобільного додатку та одночасно прихованою.

Тексти сторінок item-detail, control-master, control-detail, settings, tabs наведені в додатку В.

E:\Project\Ionic_Framework\GreenHouse\src\app\app.component.ts 21 февраля 2022 г. 16:54

```
import { Component, ViewChild } from '@angular/core';
import { SplashScreen } from '@ionic-native/splash-screen';
import { StatusBar } from '@ionic-native/status-bar';
import { TranslateService } from '@ngx-translate/core';
import { Nav, MenuController, Platform } from 'ionic-angular';

import { FirstRunPage } from '../pages';
import { User, Items } from '../providers';

@Component({
    //формуємо розмітку бічної сторінки
  template: `<ion-menu [content]="content" type="overlay">
              <ion-content class="myBg">
              <img src="assets/imgs/greenland.jpg" />
                <ion-list>
                  <button menuClose ion-item *ngFor="let p of pages" (click)="openPage(p)">
                    {{p.title | translate}}
                  </button>
                  <ion-item (click)="logoutClicked()">{{'LOGOUT' | translate}}</ion-item>
                </ion-list>
              </ion-content>
            </ion-menu>
            <ion-nav #content [root]="rootPage"></ion-nav>`,
  styles: ['.myBg,.item-md{background-color: #0a250d;color: #fff;font-size: 2.0rem;}']
})
export class MyApp {
  rootPage = FirstRunPage;

  @ViewChild(Nav) nav: Nav;
//список переходів з бічної сторінки (у нас лише один - сторінка налаштувань)
  pages: any[] = [ { title: "SETTINGS_TITLE", component: 'SettingsPage' } ]

  constructor(private translate: TranslateService, private splashScreen: SplashScreen,
              private user:User,public platform: Platform, private statusBar: StatusBar,
              private menuCtrl: MenuController, private items: Items
             ){
    platform.ready().then(() => {
      this.statusBar.styleDefault(); //встановлюємо оформлення сторінок за замовчуванням
      this.splashScreen.hide();      //приховуємо сторінку в бік
  /*заповнюємо профіл пристроїв один раз, тільки при першому запуску, обов'язково до
  процедури перевірки пароля, оскільки там вже буде інша адреса завантаження даних (url)*/
      this.items.loadProfileDev();   //процедура заповнення
    });
    this.initTranslate();            //процедура завантаження та встановлення мови програми
              }

  initTranslate() {
    this.translate.use('en');             // завантаження та встановлення англійської мови
    this.translate.use('ru');             // завантаження та встановлення російської мови
    this.translate.setDefaultLang('ua'); // завантаження та встановлення української мови
                                          // за замовчуванням
  }

  openPage(page) {                        // функція переходу на вибрану сторінку
    this.nav.setRoot(page.component);     // перехід на вибрану сторінку
  }

  logoutClicked() {              // функція процесу виходу з облікового запису користувача
    this.user.logout();                 // функція знищення облікового запису користувача
    this.menuCtrl.close();              // функція знищення історії навігації користувача
    this.nav.setRoot('LoginPage');      // перехід на сторінку авторизації
  }
}
```

1

2

Рисунок 3.13 – Зміст файлу app.component.ts

Для функціонування мобільного додатку 3-ма мовами: українською, російською та англійською, необхідно використовувати ngx-translate модуль. Встановимо до проекту плагін, далі імпортувати його в app.module.ts, app.component.ts.(рис. 3.13, підкреслене червоним та цифрою 2) (рис. 3.14, цифра 1,4).

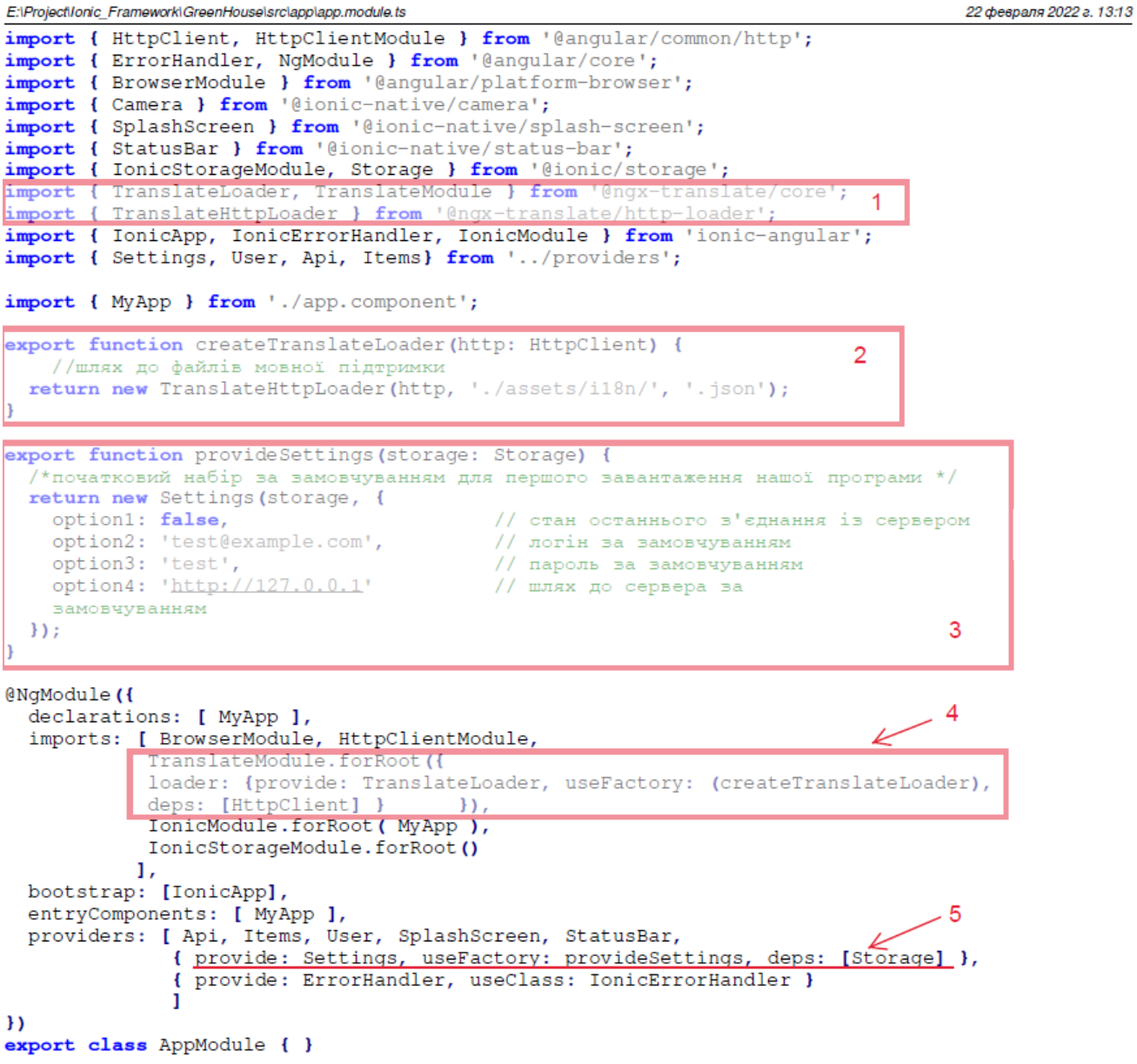


Рисунок 3.14 – Зміст файлу app.module.ts

Крім цього, до файлу app.module.ts додано функцію завантажувача з визначенням шляху до файлів перекладу та їх формат (цифра 2). Додатково в цьому же файлі прописано функцію генерування початкових значень

локальної змінної, яку використовує мобільний додаток (цифра 3) та її виконання при завантаженні як окремого провайдера (цифра 5). Прописування провайдерів дозволить їх використовувати в будь-якому модулі, протягом усього часу життєвого циклу, програмного застосунку.

Створені 3 файли перекладів, які були розміщені у папці assets (рис. 3.15). Це json файли з ключем та відповідним текстом перекладами(приклад наведено в додатку В).

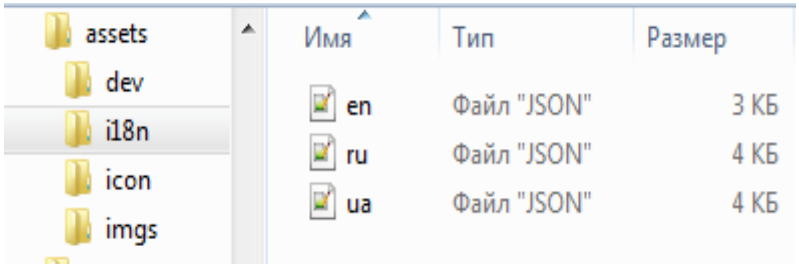


Рисунок 3.15 – json файли з трьома перекладами

Для реалізацій перекладу замінимо всі тексти у файлах сторінок мобільного додатку на функцію “translate”, яка підтягування текстів за ключами з цих файлів (рис. 3.14, цифра 2).

Початковою сторінкою мобільного додатку є сторінка login (рис. 3.17 - 3.18), яка дозволяє ввести користувачу свої значення логіну та паролю (цифра 1). Також тут є можливість обрати мову спілкування мобільного додатку (цифра 2). Процес авторизації з метою спрощення перевірки коду на початкової стадії розробки мобільного додатку та наступного тестування, мінімізується та спрощується до завдання лише отримання ідентифікаційного коду сесії доступу до даних, які надає сервер. Функціональними вимогами визначено що мобільний додаток не містить у своєму коді користувацьких даних, ні кількості параметрів або процесів, ні їх комбінацій, тому перехід на сторінку перегляду, тобто фактичне продовження роботи, без ідентифікаційного коду сесії не дасть змогу коректно зробити запит до сервера. У подальшому програмний застосунок буде додавати код сесії при

кожному запити користувацьких даних, надання яких буде залежати виключно від серверу.

E:\Project\Ionic_Framework\GreenHouse\src\pages\login\login.ts 23 февраля 2022 г. 12:26

```
import { Component } from '@angular/core';
import { TranslateService } from '@ngx-translate/core';
import { IonicPage, NavController,ToastController } from 'ionic-angular';
import { User, Settings} from '../../providers';
import { MainPage } from '../';
@IonicPage()
@Component({
  selector: 'page-login',
  templateUrl: 'login.html'
})
export class LoginPage {
                              // консолідована змінна облікового запису для форми входу.
  account: { email: string, password: string } = { email: '', password: ''};
  serv: string;               // змінна для шляху до сервера
  loginErrorString: string; // текст для користувача в разі невдалого входу в систему

  constructor(public navCtrl: NavController, public toastCtrl: ToastController,
              public user: User,public translate: TranslateService,
              private settings: Settings) { }

  ionViewCanEnter(){     //Функція виконується коли сторінка завантажена та є активною
    this.switchLang('ua');  //встановлення української мови
     this.settings.load().then(() => {           // завантаження даних з памяті
      let options = this.settings.allSettings;  // отримуємо усі значення одразу
        console.log("options",options);
        if (options.option1 == true){
        this.account.email = options.option2;    // оновлюємо значення змінних
        this.account.password = options.option3;
        }
        this.serv = options.option4;
    });
  }
    // процедура отримання дозволу на обмін даними
doLogin() {                                                                    1
  this.user.login(this.serv,this.account).subscribe((resp) => {
      //у разі успіху оновлюємо значення змінних  у локальної пам'яті
      this.settings.setValue("option1", true);
      this.settings.setValue("option2", this.account.email);
      this.settings.setValue("option3", this.account.password);
      this.navCtrl.push(MainPage); //Переходимо на сторінку перегляду параметрів
    // готуємо та виводимо повідомлення - привітання з успішною авторизацією
    let toast = this.toastCtrl.create({
      message: " Hello "+this.user._user.name ,
      duration: 3000,
      position: 'top'
    });
    toast.present();
  }, (err) => {
       //у разі невдачі оновлюємо значення змінних  у локальної пам'яті
      this.settings.setValue("option1", false);
     this.navCtrl.push(MainPage);//Переходимо на сторінку перегляду параметрів
    // готуємо та виводимо повідомлення про невдачу авторизації
    let toast = this.toastCtrl.create({
      message: this.loginErrorString,
      duration: 3000,
      position: 'top'
    });
    toast.present();
  });  }
    /* Процедура встановлення іншої мови з одночасним завантаженням повідомлення
       (про всяк випадок) у разі помилки входження до облікового запису*/          2
switchLang(l:string) {
  this.translate.use(l);
  //Отримання тексту з файлу перекладів
  this.translate.get('LOGIN_ERROR').subscribe((value) => {
    this.loginErrorString = value;
  })
}
}
```

Рисунок 3.17 – Файл login.ts

E:\Project\Ionic_Framework\GreenHouse\src\pages\login\login.html 23 февраля 2022 г. 12:36

```
<ion-content scroll="false">
<div class="splash-bg"></div>
  <div class="splash-info">
    <div class="splash-logo"></div>
    <div class="splash-intro">
      {{ 'WELCOME_INTRO' | translate }}
    </div>
  </div>
<div padding>
  <ion-title>{{ 'LOGIN_TITLE' | translate }}</ion-title>
  </div>
  <form (submit)="doLogin()" style="margin:0px 0 16px;">
    <ion-list>
      <ion-item>
        <ion-label fixed>{{ 'EMAIL' | translate }}</ion-label>
        <ion-input type="email" [(ngModel)]="account.email" name="email"></ion-input>
      </ion-item>
      <ion-item>
        <ion-label fixed>{{ 'PASSWORD' | translate }}</ion-label>
        <ion-input type="password" [(ngModel)]="account.password" name="password"></ion-input>
      </ion-item>
      <div padding>
        <button ion-button color="green" block>{{ 'LOGIN_BUTTON' | translate }}</button>
      </div>
    </ion-list>
  </form>
   <ion-footer >
  <ion-row>
    <ion-col width-25 style="text-align: right">
     <button ion-button icon-only clear (click)="switchLang('ua')"><ion-img src =
     "/assets/icon/ua.svg" alt="UA"></ion-img></button>
    </ion-col>
    <ion-col width-33 style="text-align: center">
     <button ion-button icon-only clear (click)="switchLang('en')"><ion-img src =
     "/assets/icon/eng.svg" alt="EN"></ion-img></button>
    </ion-col>
    <ion-col width-33 style="text-align: left">
     <button ion-button icon-only clear (click)="switchLang('ru')"><ion-img src =
     "/assets/icon/ru.svg" alt="RU"></ion-img></button>
    </ion-col>
  </ion-row>
 </ion-footer>
</ion-content>
```

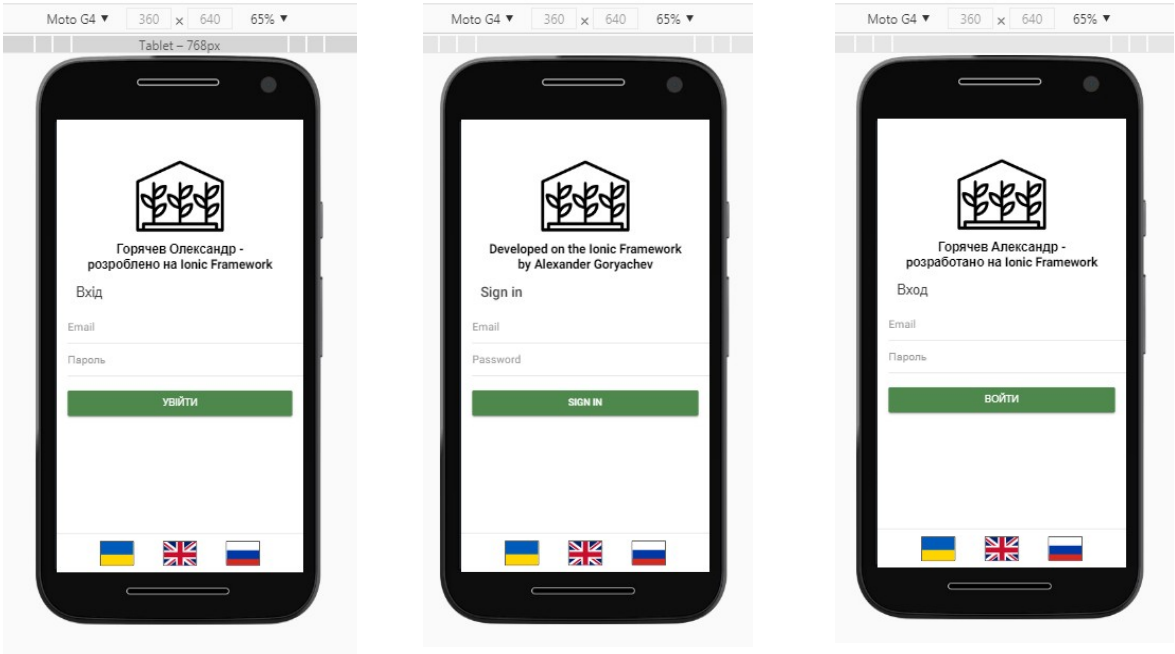
1

2

Рисунок 3.18 – Файл login.html

3.3 Тестування мобільного додатку

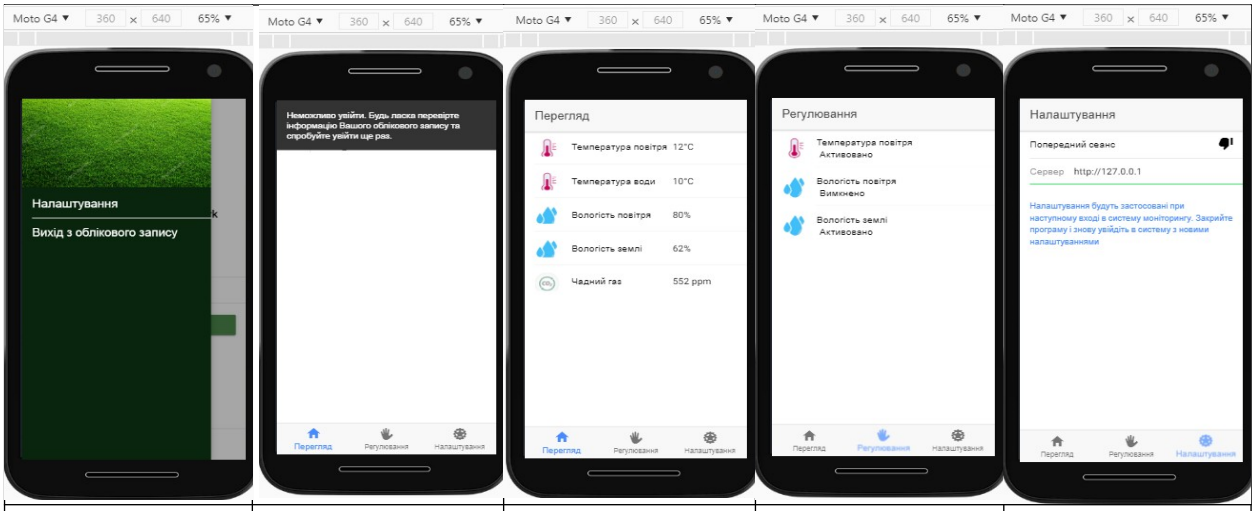
Проведемо тестування нашого мобільного додатку підтримки діяльності садової теплиці. Ми скористаємося двома способами: у настільному браузері та на різних мобільних пристроях. Виконаємо їх за часом проведення тестування та поділимо на Альфа-тестування (alpha testing) та Бета-тестування. Так як Альфа-тестування нами проводилось паралельно при створення коду мобільного додатку на симуляторі. Тому перейдемо до тестування графічного інтерфейсу мобільного додатку



а) українська мова б) англійська мова в) російська мова

Рисунок 3.19– Тестування переключення мови застосунку в браузері Chrome

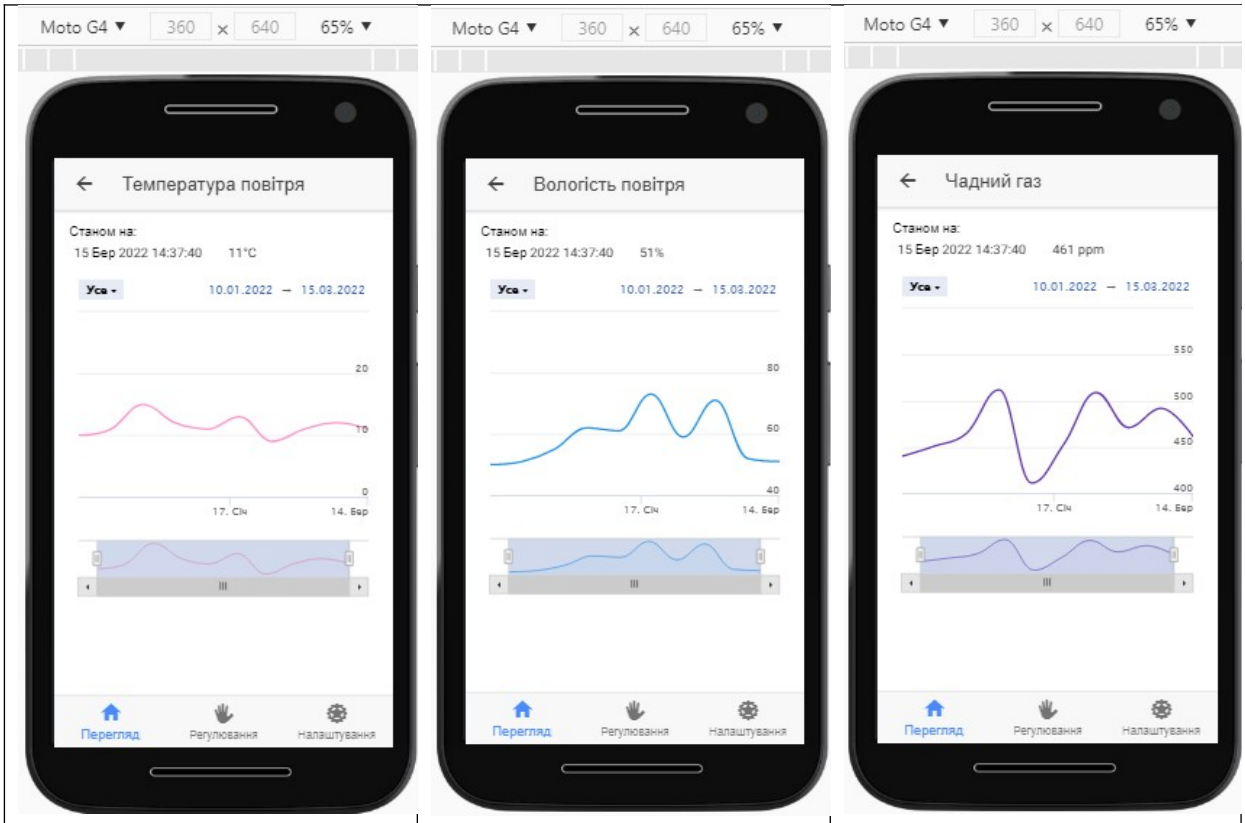
Почнемо тестування з сторінки входу до додатку та переключення мови рисунок 3.19. Далі проведемо тестування спрацювання бічної сторінки (позиція А), реагування додатку на невірні дані авторизації (позиція Б), завантаження початкової сторінки перегляду параметрів (позиція В), завантаження сторінки перегляду процесів керування (позиція Г), завантаження сторінки налаштувань застосунку (позиція Д). Підсумок цього тестування зазначено на рисунку 3.20.



а)	б)	в)	г)	д)

Рисунок 3.20 – Тестування застосунку українською мовою

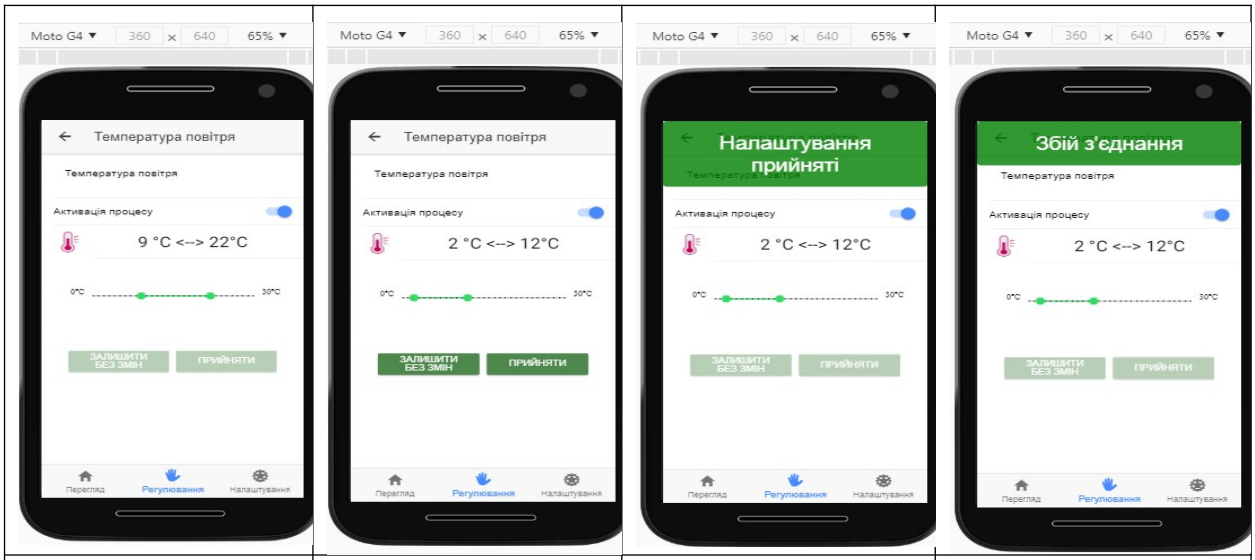
Наступне тестування (рис. 3.21) проведемо з коректного завантаження архівних даних для перегляду параметрів температури (а), вологості (б) та чадного газу (в).



а)	б)	в)

Рисунок 3.21 – Тестування завантаження архівних даних

Наступне тестування з правильного відображення стану органів керування параметрами, формування запиту до серверу на зміну регулюючих значень, отримання підтвердження прийняття обмежень тощо(рис. 3.22).



а)	б)	в)	г)

Рисунок 3.22 –Тестування сторінки зміни обмежуючих значень: а) перевірка завантаження попередніх значень; б) активація кнопок при зміні значень; в) відповідь серверу на запит щодо зміни регулюючих значень та дезактивація кнопок; г) реагування застосунку на відсутність відповіді або зв'язку з сервером, повернення програмним застосунком початкових значень.

Проведемо вибіркове тестування у симуляторі для двох операційних систем: iOS та Android(Рис.3.23)

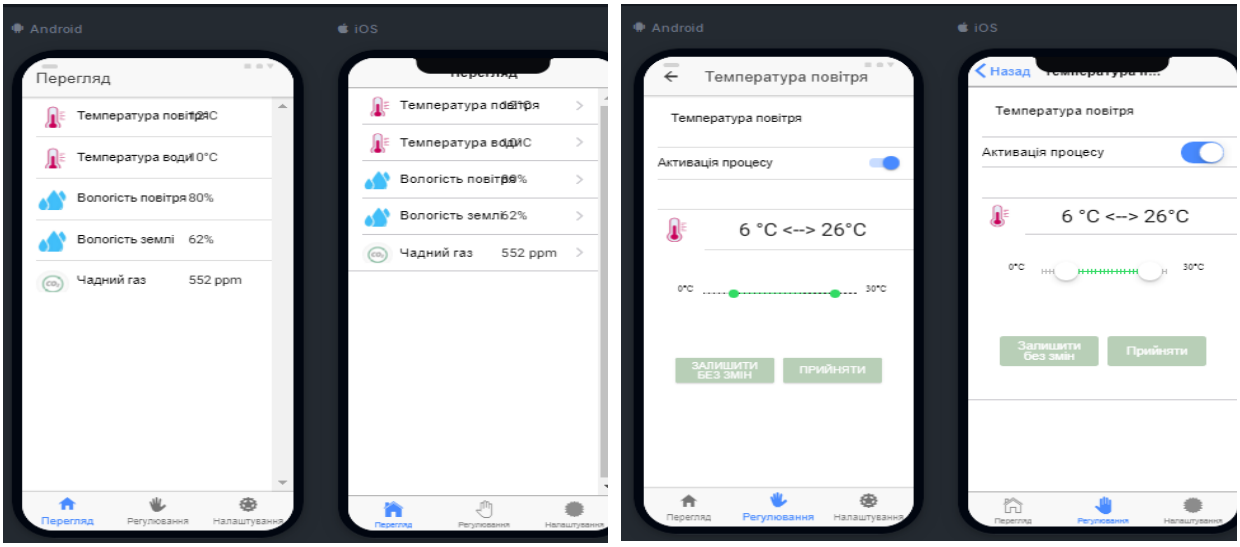


Рисунок 3.23 – Відображення мобільного додатку на платформах iOS та Android

Перейдемо до проведено закритого Бета–тестування, участь в якому приймуть запрошені особи, в нашому випадку смартфони з операційною системою Android та різними характеристиками (табл. 3.3).

Таблиця 3.3– Список смартфонів ,які залучені до тестування

Модель	Операційна система
Samsung J7	android 6
Samsung A30S	android 11
Samsung S20FE	android 12
Samsung Galaxy A32	android 11
OPPO A9	android 11

Результати тестування відображення коректно екрану в різних режимах мобільного додатку, на реальних пристроях з різною діагоналлю, матрицею та роздільною здатністю , наведені на рисунках 3.24.

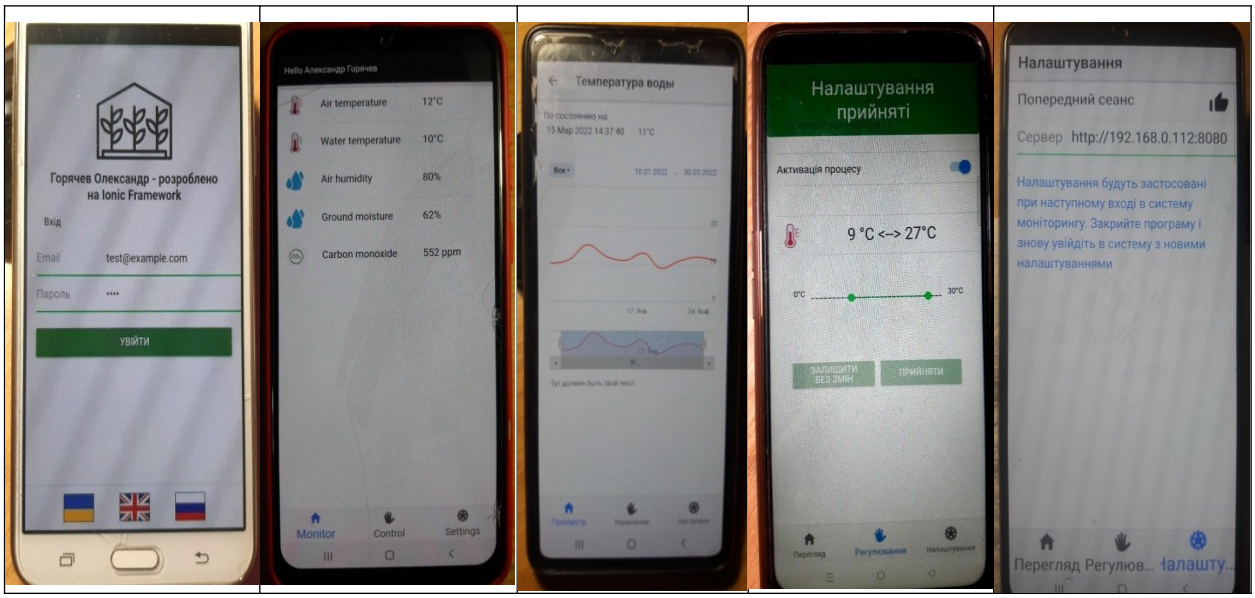
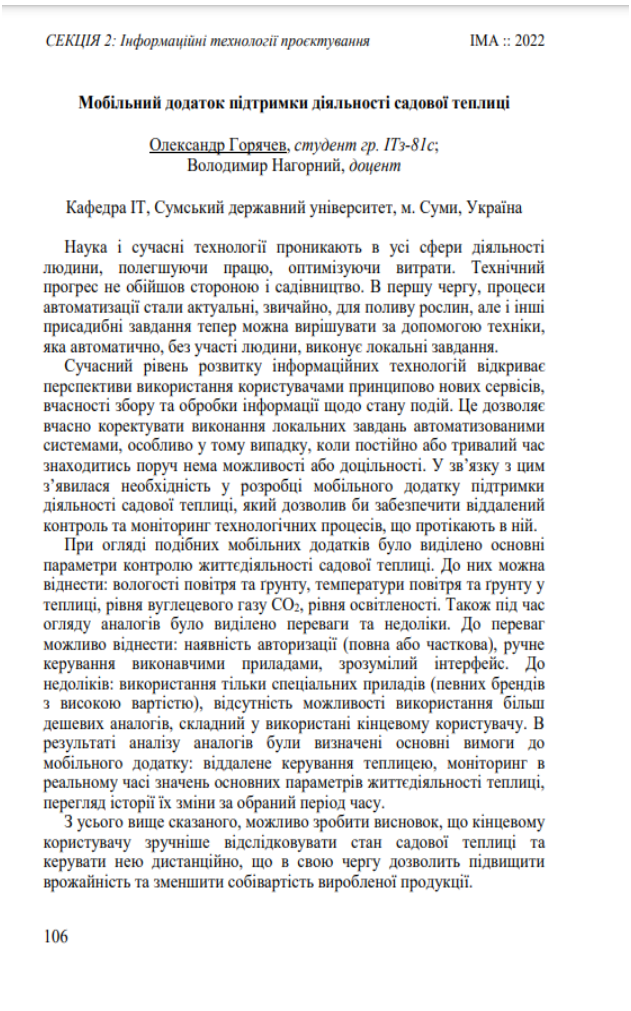


Рисунок 3.24 – Тестування додатку на смартфонах: Samsung J7, Samsung A30S, Samsung S20FE, OPPO A9, Samsung Galaxy A32

3.4 Апробація

Данна розробка була представлена на конференцій ІМА-2022 та були опубліковані тези на конференцію(рис. 3.25).



СЕКЦІЯ 2: Інформаційні технології проєктування ІМА :: 2022

Мобільний додаток підтримки діяльності садової теплиці

Олександр Горячев, *студент гр. ІТз-81с*;
Володимир Нагорний, *доцент*

Кафедра ІТ, Сумський державний університет, м. Суми, Україна

Наука і сучасні технології проникають в усі сфери діяльності людини, полегшуючи працю, оптимізуючи витрати. Технічний прогрес не обійшов стороною і садівництво. В першу чергу, процеси автоматизації стали актуальні, звичайно, для поливу рослин, але і інші присадибні завдання тепер можна вирішувати за допомогою техніки, яка автоматично, без участі людини, виконує локальні завдання.

Сучасний рівень розвитку інформаційних технологій відкриває перспективи використання користувачами принципово нових сервісів, вчасності збору та обробки інформації щодо стану подій. Це дозволяє вчасно коректувати виконання локальних завдань автоматизованими системами, особливо у тому випадку, коли постійно або тривалий час знаходитись поруч нема можливості або доцільності. У зв'язку з цим з'явилася необхідність у розробці мобільного додатку підтримки діяльності садової теплиці, який дозволив би забезпечити віддалений контроль та моніторинг технологічних процесів, що протікають в ній.

При огляді подібних мобільних додатків було виділено основні параметри контролю життєдіяльності садової теплиці. До них можна віднести: вологості повітря та грунту, температури повітря та грунту у теплиці, рівня вуглецевого газу CO_2, рівня освітленості. Також під час огляду аналогів було виділено переваги та недоліки. До переваг можливо віднести: наявність авторизації (повна або часткова), ручне керування виконавчими приладами, зрозумілий інтерфейс. До недоліків: використання тільки спеціальних приладів (певних брендів з високою вартістю), відсутність можливості використання більш дешевих аналогів, складний у використані кінцевому користувачу. В результаті аналізу аналогів були визначені основні вимоги до мобільного додатку: віддалене керування теплицею, моніторинг в реальному часі значень основних параметрів життєдіяльності теплиці, перегляд історії їх зміни за обраний період часу.

З усього вище сказаного, можливо зробити висновок, що кінцевому користувачу зручніше відслідковувати стан садової теплиці та керувати нею дистанційно, що в свою чергу дозволить підвищити врожайність та зменшити собівартість виробленої продукції.

106

Рисунок 3.25– Збірник публікацій конференцій ІМА-2022

ВИСНОВОК

В ході виконання кваліфікаційної роботи було проведено аналіз предметної області, моделювання та розробка мобільного додатку. С початку було розглянуто аналоги мобільних додатків підтримки садових теплиць. Зазначені мобільні додатки написані під конкретні апаратні можливості виконавчих механізмів та завдань агрокомпаній та фізичних осіб. На основі цього було сформовано функціональні вимоги до майбутнього мобільного додатку на основі переваг і недоліків аналогів. Це дозволило сформулювати технічне завдання, в якому показано призначення майбутнього додатку, етапи створення та вимоги до нього. Також були розглянуті методи реалізацій мобільних додатків та обраний засіб реалізації майбутнього додатку. Засобом реалізації був обраний фреймворк Ionic framework. Було сформовано перелік робіт з вказанням часових проміжків їх виконання і представлено у вигляді WBS діаграми і діаграми Ганта. Деталізували мету проекту в форматі S.M.A.R.T та розглянули майбутні ризики і стратегії їх урегулювання. При моделювання були створені контекстні діаграми в нотації IDEF та її декомпозиція IDEF0 та IDEF1. Створені діаграми варіантів використання додатку, діаграма станів та діаграма послідовності. На останньому етапі було побудована архітектура, класи додатку та розроблені макети дизайну і розроблено мобільний додаток. Після цього провели альфа і бета тестування.

В результаті отримали мобільний додаток з наступними можливостями: розподіл прав доступу між авторизованим і неавторизованим користувачем, зручний і зрозумілий інтерфейс, перегляд даних садової теплиці в реальному часі і архіву показників та внесення налаштувань в керуючий пристрій. Що цілком відповідає функціональним вимогам та поставленим задачам.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Діордієв В.Т., Кашкарьов А.О., Діордієв О.О. Автоматизована система моніторингу та керування мікрокліматом у теплиці. Матеріали науково-технічної конференції студентів та магістрантів. Науковий вісник ТДАТУ. Мелітополь, 2018. Вип. 8. Т. 2.

2. Віхрова Л.Г., Каліч В.М., Прокопенко Т.О. Адаптивна автоматизована система збору та контролю основних параметрів мікроклімату в теплиці. Збірник наукових праць Кіровоградського національного технічного університету. Техніка в сільськогосподарському виробництві, галузеве машинобудування, автоматизація. Кіровоград, 2016. Вип. 29. С. 168–172.

3. Публікація «Сучасні проблеми економіки і підприємництва. Випуск 19» 2017р.С. 20-35.

4. Інтернет-журнал про сільськогосподарські технології та точне землеробство [Електронний ресурс] – Режим доступа: https://www.agritechtomorrow.com/article/2017/12/ag-tech-decision-and-management-apps/10407

5. Програмний додаток «2Agrocloud» (Провайдер LLC Data House) Режим доступу:

https://play.google.com/store/apps/details?id=com.app.agrocloud, https://apps.apple.com/ru/app/2agrocloud/id1497502245

6. Програмний додаток «Розумна теплиця» (Меремьянин Сергей)

Режим доступу:

https://play.google.com/store/apps/details?id=com.embarcadero.AcsMA&hl=ru&gl=US

7. Програмний додаток «Розумна теплиця» [7]. Режим доступу:

https://play.google.com/store/apps/details?id=com.impuls.teplitsa&hl=gsw

8. Програмний додаток «Моніторинг і управління теплицею з телефону або планшета З ОС ANDROID» (аматорська розробка для малого обсягу рослин та локальна реалізація, Виктор Петин). Режим доступу:

https://playarduino.ru/uroki-arduino/upravlenie-teplitsej-s-telefona/

9. Програми модель MVC

https://hi-news.pp.ua/kompyuteri/14628-programa-model-view-controller-mvc-scho-ce-osoblivost-opis.html

10. Метод Scrum [Електронний ресурс] – Режим доступа:

https://brainrain.com.ua/uk/scrum-upravlinnya-proektom/

11. *Rapid application development* [Електронний ресурс] – Режим доступа:

https://kissflow.com/low-code/rad/rapid-application-development/

12. *Rational Unified Process* [Електронний ресурс] – Режим доступа:

http://www.interface.ru/rational/rup01_t.htm

13.Офіційний сайт Ionic Framework [Електронний ресурс] – Режим доступа: https://ionicframework.com/

14. Ярушкіна, Н. Проектування баз даних для економічних інформаційних систем в середовищі сервера даних ORACLE 7.3 / Н. Ярушкіна, Т. МЕРКУЛОВА. - Ульяновськ: УлГТУ, 2002. - 160 с

15. Методология IDEF0 [Електронний ресурс] – Режим доступу до ресурсу: https://itteach.ru/bpwin/metodologiya-idef0

16. Архітектурний шаблон MODEL-VIEW-CONTROLLER (MVC) для побудови класів [Електронний ресурс] Режим доступу: https://schoolboyprog10.blogspot.com/p/mvc-mvc-model-view-controller.html

17. AppMaster.io [Електронний ресурс] Режим доступу: https://studio.appmaster.io

18. Розробка кроссплатформенного мобільного додаткуза допомогою Ionic Framework [Електронний ресурс]. - Режим доступу до матеріалу: https://habrahabr.ru/post/255653/

ДОДАТОК А Технічне завдання

1. Призначення й мета створення мобільного додатку

1.1 Призначення мобільного додатку

Мобільний додаток має надавати у повному обсязі інформацію користувачу про стан садової теплиці, управління системою віддалено.

1.2 Мета створення мобільного додатку

Метою проекту є розробка мобільного додатку підтримки діяльності садової теплиці, за допомогою якого користувач зможе спостерігати за показниками садової теплиці у реальному часі та збереженими в архів даними, а також вносити зміни в керуючий прилад.

1.3 Цільова аудиторія

До цільової аудиторії мобільного додатку можна віднести всіх людей, що займаються фермерством в садових теплицях, завдяки зручному і доступному інтерфейсу.

2 Вимоги до мобільного додатку

2.1 Вимоги до мобільного додатку в цілому

2.1.1 Вимоги до структури й функціонування мобільного додатку

Мобільний додаток має розміщуватися в Google Play і App Store .
Мобільний додаток повинен виконувати:

– вхід до облікового запису шляхом авторизації на початку роботи додатку;

– обрання потрібної теплиці, якщо їх декілька;

– перегляд списку характеристик та значення параметрів характеристик системи підтримки діяльності теплиці;

– перегляд архівних значень параметрів характеристик системи підтримки діяльності теплиці у вигляді графіку;

– редагування налаштувань керуючих параметрів системи підтримки діяльності теплиці шляхом вибору дискретних значень з фіксованого діапазону;

– вибір мови додатку.

2.1.2 Вимоги до персоналу

Персонал повинен володіти базовими навичками користування базами даних MySQL, а саме внесення даних авторизації.

2.1.3 Вимоги до збереження інформації

Уся інформація надана у мобільному додатку буде зберігатися у базі даних реалізованій засобами системи управління базами даних MySQL.

2.1.4 Вимоги до розмежування доступу

Доступ до мобільного додатку здійснюється за особистим логіном і паролем.

Відповідно до прав доступу до інформації у мобільному додатку, усіх користувачів можна поділити на користувача та адміністратора.

Користувачі можуть переглядати усі сторінки мобільного додатку, ознайомитись з моніторингом (станом) теплиці та встановлювати регулюючи обмеження.

2.2 Структура мобільного додатку

2.2.1 Загальна інформація про структуру мобільного додатку

Структура мобільного додатку – набір розміток графічного інтерфейсу.

Сторінка авторизації – користувач вводить данні авторизації, а також може обрати мову.

Перегляд – відображення стану датчиків.

Регулювання – налаштування роботи регулюючих програм.

Налаштування – можна редагувати налаштування мобільного додатку.

2.2.2 Навігація

Меню повинне бути розташоване в низу екрану. Меню необхідне для переміщення користувача по всім доступним розміткам. Воно повинне відображатися внизу екрану, щоб користувач міг перейти на будь-яку сторінку мобільного додатку.

2.2.3 Наповнення мобільного додатку (контент)

Заповнення та редагування контенту мобільного додатку має бути зроблено через сторінки з елементами керування, використовуючи інформацію з бази даних.

Вся інформація для наповнення мобільного додатку надходить з бази даних .

2.2.4 Дизайн та структура додатку

Стиль мобільного додатку має бути виконаний сучасним, приємним для сприйняття, у якості основних кольорів обрано зелені та білі кольори.

2.2.5 Система навігації (карта мобільного додатку)

Карта мобільного додатку зображена на рисунку А.1.

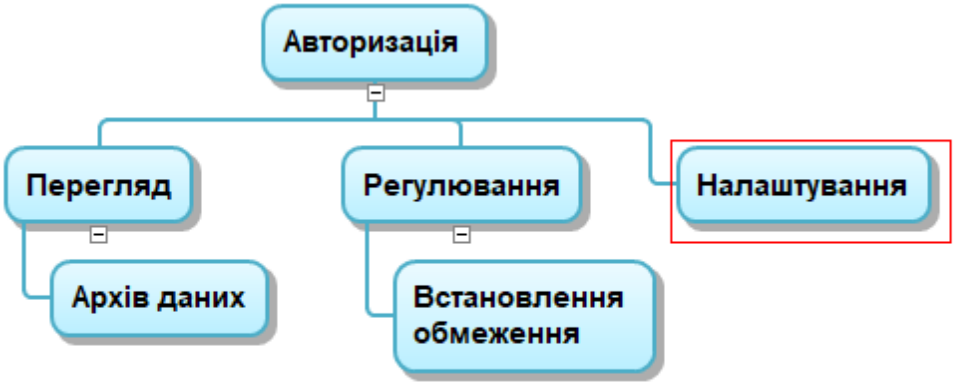


Рисунок А.1 – Карта мобільного додатку

2.3 Вимоги до функціонування системи

2.3.1 Потреби користувача

Потреби користувача, визначені на основі рішення замовника, представлені у таблиці А.1.

Таблиця А.1 – «Потреби користувача»

UN ID	Опис потреби (User Need)	Джерело (Source)
UN-100	Можливість передивлятись поточний стан та значення параметрів датчиків температури і вологості,термостатів, терморегуляторів та інших виконавчих та контролюючих приладів та перегляд архівних значень	Користувач
UN-200	Можливість введення обмежень, щодо роботи виконавчих приладів (Керувати мікрокліматом теплиці) та налаштувань додатку	Користувач

2.3.2 Функціональні вимоги

На основі потреб користувача були визначені такі функціональні вимоги.

Таблиця А.2 – «Функціональні вимоги»

UR ID	Назва UR	Опис	Посилання на UN
UR-STK-100	Перегляд списків характеристик теплиці	Групування по розділам функціональних завдань системи, а саме перегляд, управління, настройки. Перегляд списків характеристик теплиці кожного розділу в меню	UN-0100, UN-0200
UR-STK-200	Видача параметрів теплиці за вибраною характеристикою	Відображення детальної інформації про вибрану характеристику , саме поточне значення, за наявності архів значень тощо;	UN-0100
UR-STK-300	Регулювання параметрів роботи виконавчих приладів	Введення обмежень параметрів характеристик щодо роботи виконавчих приладів	UN-0200
UR-STK-400	Портретна орієнтація екрану	Додаток підтримує можливість роботи в портретній орієнтації екрану	UN-0100, UN-0200
UR-STK-500	Мова додатку	Додаток повинен функціонувати 3-ма мовами: українською, російською та англійською версією користувальницького інтерфейсу	UN-0100, UN-0200

2.3.3 Системні вимоги

В цьому розділ визначаються і розподіляються та вказуються на системні вимоги, визначені розробником. Їх перелік наведений в таблиці А.3.

Таблиця А.3 – «Системні вимоги»»

ID	Назва SR	SR	Пріоритет	Метод верифікації	UR	Статус
1	2	3	4	5	6	7
SRS-0100	Портретна орієнтація екрану	Система забезпечую роботу додатка тільки в портретній орієнтації екрану	H	M	UR-STK-400	
SRS-0200	Авторизація	Система повинна здійснювати авторизацію користувачів із правами на роботу з додатком. Після успішної авторизації система повинна перевести користувача на початкову сторінку.	B	T	UR-STK-100, UR-STK-200(300),	В роботі
SRS-0300	Завершення роботи	Система у верхньої частини екрану виводить кнопку виходу з додатку та забезпечує завершення авторизованої сесії	B	M	SRS-0200	
SRS-0400	Вибір розділу	Система у нижній частини екрану виводить меню з пунктами розділів: «Перегляд», «Управління» , «Настройки». Початковою сторінкою при запуску додатку є розділ «Перегляд». Перехід від одного списку до іншого здійснюється натисканням на вибір меню розділів	C	T	UR-STK-400	В роботі
SRS-0500	Показ характеристик	Система виводить на сторінці вибраного розділу список характеристик теплиці Кожен елемент списку - посилання на інший екран з його описом	C	M	UR-STK-0100, UR-STK-0200	
SRS-0600	Показ значень обраної характеристики	Система повинна показувати користувачу поточне значення (останнє) параметру обраний характеристики у нової сторінці	C	M	UR-STK-0100, UR-STK-0200	
SRS-0700	Архівні значення параметрів	Система повинна для розділу «Перегляд», при показі параметрів, забезпечувати показ архівних значеннь параметру характеристики у вигляді графіку.	H	M	UR-STK-200	
SRS-0800	Регулювання значення параметрів	Система повинна для розділу «Управління», при показі параметрів, забезпечувати показ елементів регулювання параметру характеристики у вигляді фіксованого вибору значень.	B	T	UR-STK-300	В роботі
SRS-0900	Застосування нових регулюючих значень	Система повинна після закінчення налаштувань регулюючих обмежень надати користувачу змогу відмовитись або прийняти нові значення діапазону (повинен зробити обов’язковий вибір)	B	T	UR-STK-300	В роботі

Продовження таблиці А.3

ID	Назва SR	SR	Пріоритет	Метод верифікації	UR	Статус
1	2	3	4	5	6	7
SRS-1000	Мова додатку	Система забезпечує функціонування додатку 3-ма мовами: українською, російською та англійською версією користувальницького інтерфейсу	Н	Т	UR-STK-0500	В роботі
SRS-1100	Побудова графіків	Система забезпечує побудову графіків не повільніше ніж за 2 секунди	Н	М	UR-STK-200	
SRS-1200	Контроль введеної інформації	Система забезпечує контроль введеної інформації та блокування некоректних дій користувача при роботі з системою при авторизації.	С	Т	UR-STK-100	В роботі
SRS-1300	Технології	Для реалізації програми використовувати мову JavaScript (TypeScript) та фреймворк Ionic 2	Н			Проект

Примітки: В – високий; С – середній; Н – низький; М – моніторинг; Т – тест.

2.4 Вимоги до видів забезпечення

2.4.1 Вимоги до інформаційного забезпечення

Реалізація мобільного додатку відбувається з використанням Typescript.

2.4.2 Вимоги до лінгвістичного забезпечення

Мобільний додаток має бути виконаний українською мовою.

2.4.3 Вимоги до програмного забезпечення

Програмне забезпечення користувача повинне задовольняти наступним вимогам:

– Android 4 та більше;
– IOS 10.0 та більше.

3 Склад і зміст робіт зі створення мобільного додатку

Докладний опис етапів роботи зі створення мобільного додатку наведено в таблиці А.3.

Таблиця А.3 – «Етапи створення мобільного додатку»

№	Склад і зміст робіт	Строк розробки (у робочих днях)
1	Постановка цілей необхідних для досягнення певного результату	2 день
2	Складання технічного завдання	4 дні
3	Підготовка прототипу	3 дні
4	Створення макету дизайну мобільного додатку	3 дні
5	Верстка	2 дні
6	Робота над модулями для мобільного додатку	3 дні
7	Робота з контентом	2 день
8	Розміщення контенту та каталогів з фото у мобільного додатку	2 день
9	Перевірка працездатності мобільного додатку	2 день
10	Завершення роботи	2 день
	Загальна тривалість робіт	25 днів

4. Вимоги до складу й змісту робіт із введення мобільного додатку в експлуатацію

Для того, щоб мобільним додатком могли користуватися користувачі необхідно встановити додаток на пристрій користувача.

ДОДАТОК Б Планування робіт

Наука і сучасні технології проникають в усі сфери діяльності людини, полегшуючи працю, оптимізуючи витрати. Технічний прогрес не обійшов стороною і садівництво. В першу чергу, процеси автоматизації стали актуальні, звичайно, для поливу рослин, але й інші присадибні завдання тепер можна вирішувати за допомогою техніки. Тому вдосконалення процесів функціонування садової теплиці позитивно впливатиме не тільки на збільшення продуктивності, а й зручність керування віддалено від ней.

Деталізація мети проекту методом SMART. Щоб проект був успішним та конкурентоспроможним треба на концептуальному етапі правильно визначити його мету за допомогою SMART–методу. Результати деталізації методом SMART розміщені у таблиці Б.1.

Таблиця Б.1 – «Деталізація мети проекту методом SMART»

Specific (конкретна)	Створити мобільний додаток для організації діяльності садової теплиці
Measurable (вимірювана)	Розробити мобільний додаток за 6 місяці, використовуючи мінімум ресурсів.
Achievable (досяжна, узгоджена)	Для виконання проекту наявні необхідні знання HTML, CSS, мови програмування JavaScript, PHP, середа розробки Apache Cordova та навичок написання документації. Враховуючи доступні ресурсні можливості та обмеження мета є такою, яку можливо досягти.
Relevant (реалістична)	Створений мобільний додаток садової теплиці наддасть комфортну можливість спостереження за садовою теплицею.
Time-framed (обмежена в часі)	Ціль має часове обмеження. Термін досягнення мети проекту визначено за домовленістю між замовником та виконавцем і складає шість місяців.

Планування змісту робіт. WBS (Work Breakdown Structure – Ієрархічна структура робіт) – це графічний вигляд елементів проекту, які згруповані ієрархією у єдине ціле з продуктом проекту. Структура декомпозиції робіт

орієнтована на досконале виконання робіт по частинам і сама є ключовою частиною проекту, яка спрямована на організацію командної роботи. Елементами декомпозиції можуть бути продукти, дані та послуги. Більше того, WBS забезпечує необхідним каркасом для ретельної оцінки термінів, а також контролю та графіків роботи.

На найвищому (першому) рівні розміщений продукт проекту. Основні дії та заходи, що забезпечують досягнення мети проекту, зафіксовані на другому рівні декомпозиції. Декомпозиція робіт виконується до тих пір, поки вони не стануть елементарними (простими).

Елементарні роботи – це дії, які мають однозначний чіткий результат, на які призначена відповідальному одна конкретна особа, для якої можна обчислити витрати праці і тривалість виконання. На рисунку Б.1 представлено WBS з розробки мобільний додатку організації діяльності садової теплиці.

Планування структури виконавців. Наступним етапом після декомпозиції процесів є розробка організаційної структури виконавців або OBS, яка визначається як графічна структура відображення учасників або відповідальних осіб, які беруть участь у реалізації проекту.

У ролі відповідальних осіб виступають співробітники, що відповідають за організацію і виконання елементарної роботи, що зазначена у WBS. Кожну елементарну роботу можна розглядати як окремий проект.

На рисунку Б.2 представлено організаційну структуру планування проекту. Список виконавців, що функціонують в проекті описано в таблиці Б.2.

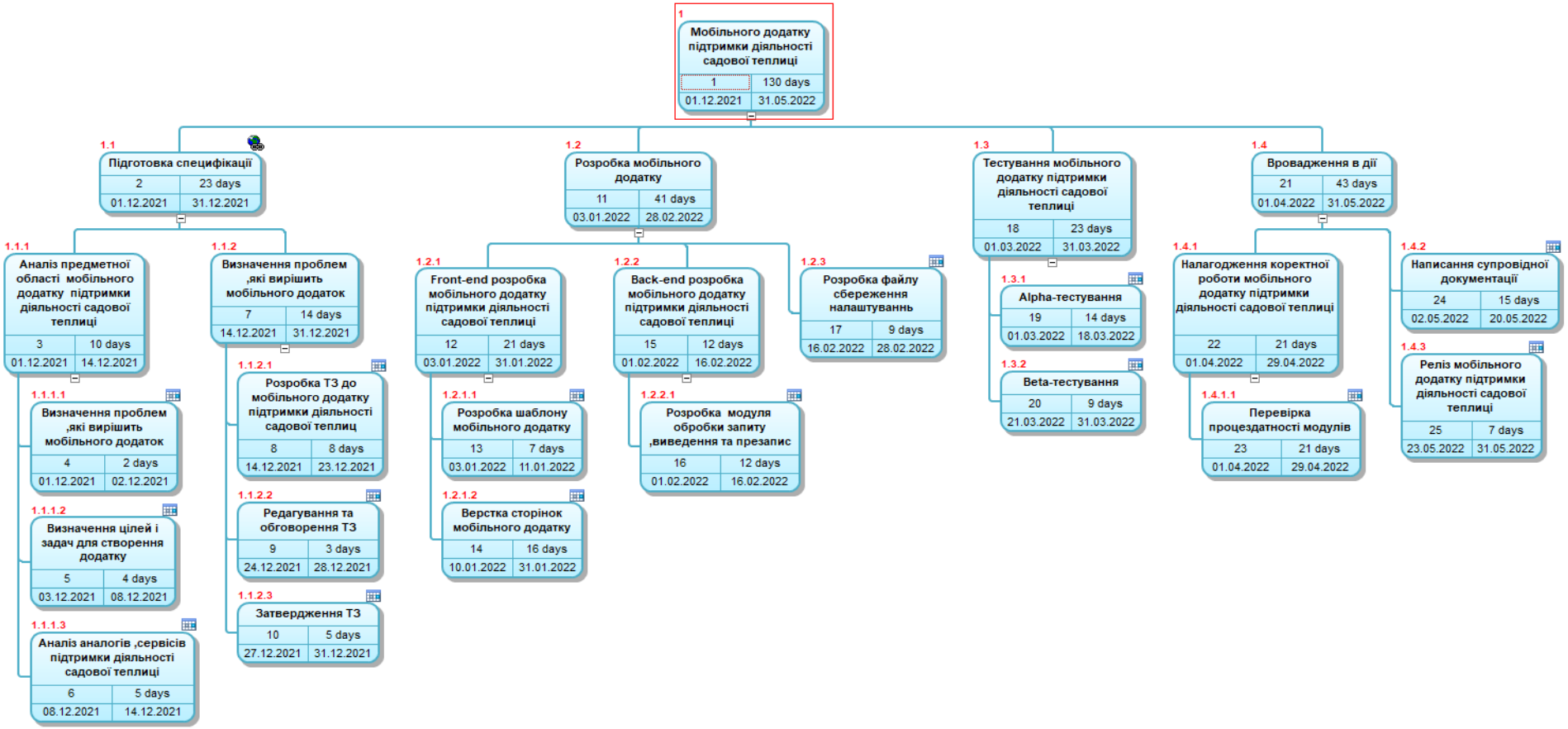


Рисунок Б.1 – WBS-структура робіт проекту

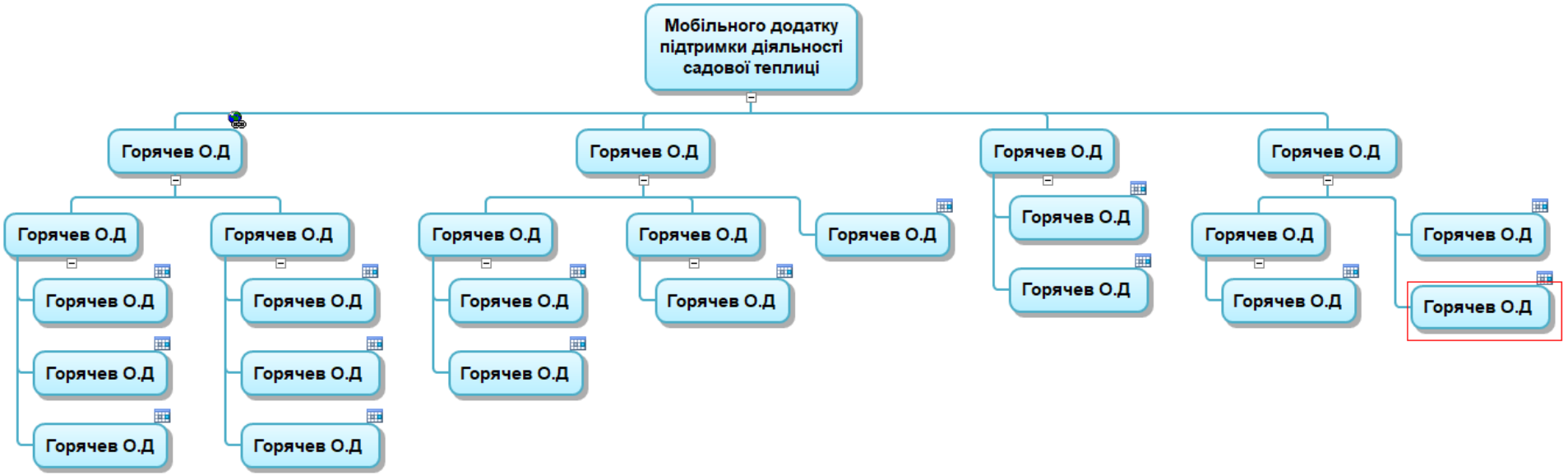


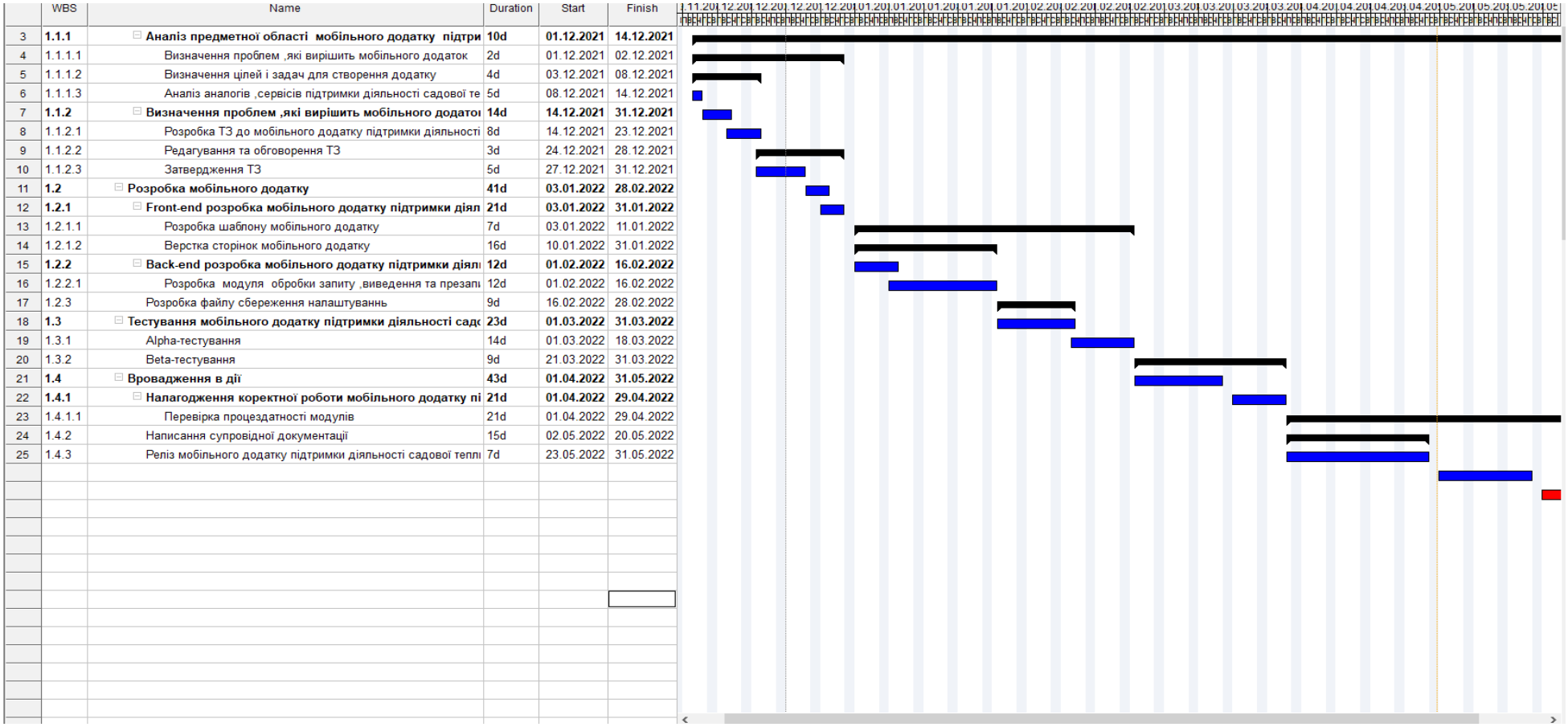
Рисунок Б.2 – OBS-структура робіт проект

Таблиця Б.2 – Виконавці проекту

Роль	Ім’я	Проектна роль
Розробник	Горячев О.Д.	Виконує front-end та back-end розробку Проектувальник
Проектувальник	Горячев О.Д.	Виконує проектування бази даних та розробляє структуру мобільний додатку.
Тестувальник	Горячев О.Д.	Відповідає за тестування функціоналу та дизайну мобільний додатку.
Керівник проекту	Горячев О.Д.	Формує завдання на розробку проекту.
Менеджер проекту	Горячев О.Д.	Відповідає за виконання термінів, розподіл ресурсів та завдань між учасниками. Виконує збір та аналіз даних.

Діаграма Ганта. Побудова календарного графіку (діаграми Ганта) є одним з важливих етапів планування проекту, що виглядає як розклад виконання робіт з реальним розподілом дат. Завдяки йому можна отримати достовірне уявлення про тривалість процесів з обмеженнями у ресурсах, урахуванням вихідних днів та свят.

Календарний графік проекту представлено на рисунку Б.3



	WBS	Name	Duration	Start	Finish
3	**1.1.1**	**Аналіз предметної області мобільного додатку підтри**	**10d**	**01.12.2021**	**14.12.2021**
4	1.1.1.1	Визначення проблем ,які вирішить мобільного додаток	2d	01.12.2021	02.12.2021
5	1.1.1.2	Визначення цілей і задач для створення додатку	4d	03.12.2021	08.12.2021
6	1.1.1.3	Аналіз аналогів ,сервісів підтримки діяльності садової те	5d	08.12.2021	14.12.2021
7	**1.1.2**	**Визначення проблем ,які вирішить мобільного додатоі**	**14d**	**14.12.2021**	**31.12.2021**
8	1.1.2.1	Розробка ТЗ до мобільного додатку підтримки діяльності	8d	14.12.2021	23.12.2021
9	1.1.2.2	Редагування та обговорення ТЗ	3d	24.12.2021	28.12.2021
10	1.1.2.3	Затвердження ТЗ	5d	27.12.2021	31.12.2021
11	**1.2**	**Розробка мобільного додатку**	**41d**	**03.01.2022**	**28.02.2022**
12	**1.2.1**	**Front-end розробка мобільного додатку підтримки діял**	**21d**	**03.01.2022**	**31.01.2022**
13	1.2.1.1	Розробка шаблону мобільного додатку	7d	03.01.2022	11.01.2022
14	1.2.1.2	Верстка сторінок мобільного додатку	16d	10.01.2022	31.01.2022
15	**1.2.2**	**Back-end розробка мобільного додатку підтримки діял**	**12d**	**01.02.2022**	**16.02.2022**
16	1.2.2.1	Розробка модуля обробки запиту ,виведення та презап	12d	01.02.2022	16.02.2022
17	1.2.3	Розробка файлу збереження налаштуваннь	9d	16.02.2022	28.02.2022
18	**1.3**	**Тестування мобільного додатку підтримки діяльності сад**	**23d**	**01.03.2022**	**31.03.2022**
19	1.3.1	Alpha-тестування	14d	01.03.2022	18.03.2022
20	1.3.2	Beta-тестування	9d	21.03.2022	31.03.2022
21	**1.4**	**Вровадження в дії**	**43d**	**01.04.2022**	**31.05.2022**
22	**1.4.1**	**Налагодження коректної роботи мобільного додатку пі**	**21d**	**01.04.2022**	**29.04.2022**
23	1.4.1.1	Перевірка процездатності модулів	21d	01.04.2022	29.04.2022
24	1.4.2	Написання супровідної документації	15d	02.05.2022	20.05.2022
25	1.4.3	Реліз мобільного додатку підтримки діяльності садової тепл	7d	23.05.2022	31.05.2022

Рисунок Б.3 – Календарний графік проект

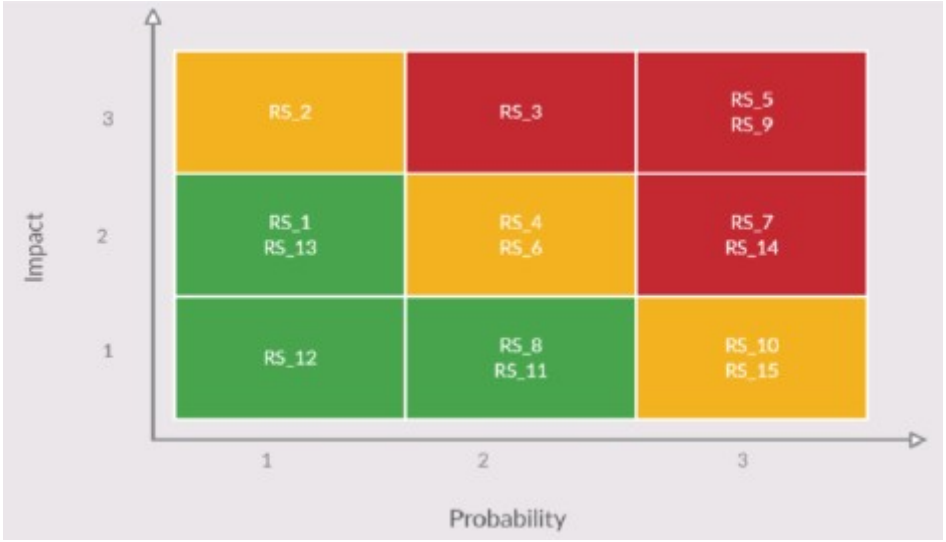
Управління ризиками проекту. Під час виконання якісної оцінки ризиків треба визначити ризики, які мають бути усунені якнайшвидше. В залежності від ступеня важливості ризику - реагування буде відповідне. Наступним етапом є виконання кількісного оцінювання ризиків. Кількісне та якісне оцінювання можуть виконувати одночасно або окремо, що залежить від ступеня забезпечення проекту. У таблиці Б.3 представлено шкалу для класифікації ризиків за величиною впливу на проект та ймовірністю виникнення.

Таблиця Б.4 – «Шкала оцінювання ризиків за ймовірністю виникнення та величиною впливу»

Оцінка	Ймовірність виникнення	Вплив ризику	Тип ризику
1	Низька	Низький	Прийнятні
2	Середня	Середній	Виправдані
3	Висока	Високий	Недопустимі

Для того, щоб знизити негативний вплив ризиків на проект треба виконати планування реагування на них. До нього входить визначення ефективності розробки та оцінка наслідків впливу на проект. Оцінювання виконується за показниками, що описані в таблиці Б.5. У результаті планування реагування було отримано матрицю ймовірності виникнення ризиків та впливу ризику, що зображена на рисунку Б.6.

Зеленим кольором на матриці позначають прийнятні ризики, жовтим – виправдані, а червоним – недопустимі.



– Рисунок Б.6. Матриця ймовірності

Класифікація ризиків за рівнем, відповідно до отриманого значення індексу, представлена у таблиці Б.4. У таблиці Б.5 описано ризики та стратегії реагування на кожен з них.

Таблиця Б.5 – «Шкала оцінювання за рівнем ризику»

№	Назва	Межі	Ризики, які входять(номера)
1	Прийнятні	$1 \leq R \leq 2$	1,8,11,12,13
2	Виправдані	$3 \leq R \leq 4$	2,4,6,10,15
3	Недопустимі	$6 \leq R \leq 9$	3,5,7,9,14

Таблиця Б.5 – «Ризики та стратегії реагування»

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг ризику	План А	Тип стратегії реагування	План Б
RS_1	Відкритий	Непорозуміння між розробником та замовником	Низька	Середній	3	1. Налагодити гарні відносини між розробником та керівником. 2. Дотримуватися ділового етикету спілкування. 3. Створити комфортні умови для співпраці	Попередження	При виявлення непорозуміння потрібно вияснити, що саме стало причиною непорозуміння обсудити її та створити здорову атмосферу в колективі.
RS_2	Відкритий	Поява альтернативного продукту	Низька	Високий	3	1.Провести попереднє дослідження альтернативних продуктів. 2. Вибрати унікальну стратегію створення проекту.	Прийняття	
RS_3	Відкритий	Нечітке завдання на розробку	Середня	Високий	6	1.Ясно і однозначно обговорити із замовником усі види вимог. 2.Скласти глосарій для запобігання розбіжностей у розумінні слів та термінів. 3.Періодичний контроль замовником етапів роботи.	Попередження	При виявленні невідповідносте деяких характеристик продукту заявленим вимогам потрібно уважно та чітко окреслити те, що було виконано невірно та зробити правки

Продовження таблиці Б.5

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг ризику	План А	Тип стратегії реагування	План Б
RS_4	Відкритий	Низька кваліфікація розробників	Середня	Середній	4	1.Підвищити кваліфікацію персоналу. 2.Переглянути онлайн-ресурси для підвищення рівня знань.	Пом’якшення	Врахувати час на підготовку працівників. Видати літературу, переглянути онлайн-уроки
RS_5	Відкритий	Неоптимальний розподіл часу	Висока	Високий	9	Провести аналіз актуальності найважливіших процесів та робіт. Звернути особливу увагу на правильність розподілу часу. Правильно визначити пріоритети виконання робіт. Чітко дотримуватися календарного плану.	Пом’якшення	Змінити порядок пріоритетів робіт. Знайти способи оптимізації роботи з вже існуючою розстановкою. Обговорити варіанти внесення поправок до термінів реалізації із замовником.

Продовження таблиці Б.5

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг ризику	План А	Тип стратегії реагування	План Б
RS_6	Відкритий	Часте внесення змін у ТЗ	Середня	Середній	4	1.Виділити всі необхідні параметри проекту 2.Чітко описати вимоги до проекту. 3.Обговорити всі технічні засоби виконання проекту та умови реалізації	Перенесення	Узгодити всі положення з замовником, у разі потреби внести необхідні зміни та поправки.
RS_7	Відкритий	Вибір не ефективної технології розробки	Середня	Високий	6	1.Проаналізувати методи та засоби, для виконання проекту. 2.Обрати зрозумілу та легку в використанні технологію розробки.	Пом’якшення	Виділити час та ресурсі на пошуки покращення обраної технології. Застосувати допоміжні ресурси.
RS_8	Відкритий	Неправильна оцінка в масштабі проекту	Низька	Середній	2	1.Провести детальний аналіз проекту. 2.Визначити основні етапу проекту, розподілити час на їх виконання. 3.Проаналізувати масштаби проекту на основі додаткових джерел.	Пом’якшення	Переоцінка масштабів проекту. Перебудова стратегії реалізації проекту

Продовження таблиці Б.5

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг ризику	План А	Тип стратегії реагування	План Б
RS_9	Відкритий	Помилки проектування	Висока	Високий	9	На етапі проектування тісно співпрацювати із замовником та на певних етапах демонструвати поточні результати	Пом’якшення	Здійснювати проміжний контроль результатів в ході виконання проекту.
RS_10	Відкритий	Збої в роботі програмного забезпечення	Низька	Високий	3	1.Підготувати резерв програмних засобів. 2.Залучити спеціаліста для усунення збоїв.	Попередження	Замінити програмне забезпечення
RS_11	Відкритий	Відсутність резервних копій даних	Низька	Середній	2	1.Налаштувати автоматичне збереження даних. 2.Зберігати дані на різних носіях інформації.	Попередження	Робити копію даних після кожного виконаного етапу
RS_12	Відкритий	Реалізація непотрібного функціоналу	Низька	Низький	1	Попередити замовника про можливість додаткового функціоналу.	Використання	Обговорити вигоди і збитки від можливих змін проекту

Продовження таблиці Б.5

RS_13	Відкритий	Невиконання моніторингу проекту	Середня	Низький	2	Здійснювати проміжний контроль результатів в ході виконання проекту. Здійснювати моніторинг проекту працівниками.	Перенесення	Здійснювати моніторинг проекту замовником. Надання проміжних результатів виконання проекту після кожного етапу.
RS_14	Відкритий	Виникнення проблем із програмним забезпеченням користувачів	Середня	Високий	6	1.Розробка проекту з урахуванням вимог до програмного забезпечення користувачів проекту. 2.Модифікація проекту з урахуванням різних версій програмного забезпечення, яке буде застосовуватися.	Прийняття	
RS_15	Відкритий	Зміна вимог замовника в процесі розробки проекту	Низька	Високий	3	Узгодити всі питання на початкових етапах, щоб мінімізувати кількість змін під час розробки	Пом’якшення	Переоцінка проекту, кожного разу, коли вимоги змінюються

ДОДАТОК В Лістинг коду

Файл item-detail.ts

```
import { Component } from '@angular/core';
import { IonicPage, NavController, NavParams, ToastController } from 'ionic-angular';
import { Items, Api} from '../../providers';
import * as Highcharts from "highcharts/highstock";//ініціалізація модулів stock tools
import { TranslateService } from '@ngx-translate/core';

@IonicPage()
@Component({
  selector: 'page-item-detail',
  templateUrl: 'item-detail.html'
})
export class ItemDetailPage {
// ініціалізація  змінних
  item: any;  last_times: any; last_value: any; json_data: any; shortMonth: any;
  serverErrorString: string; selectorZoomString: string;


constructor(public navCtrl: NavController, public navParams: NavParams, public items: Items, public api: Api,private translate: TranslateService, public toastCtrl: ToastController ) {  this.item = navParams.get('item') || items.defaultItem; }
//завантажуеться після ініціалізації сторінки
 ngAfterViewInit ( ) {
       //встановлення мови меток на сторінці
        this.translate.get('SHORTMONTHS').subscribe((value) => {this.shortMonth = value;});
                  this.translate.get('SERVER_STATUS').subscribe((value) => {this.serverErrorString = value;});
                  this.translate.get('SELECTOR_ZOOM').subscribe((value) => {this.selectorZoomString = value;});
       //встановлення шаблону запиту та завантаження даних
const my_url = 'index.php?dev='+this.item.nicname;
this.api.get(my_url).subscribe(data => {if (data != null) {
              this.json_data=data;              this.last_times=this.timeConverter( data[this.json_data.length-1][0] );
 this.last_value = data[this.json_data.length-1][1];
 this.fetchData(data); //побудова графіку
}else{ this.Toaster(this.serverErrorString[1]);}//вивід повідомлень відповідно до отриманої помилки
}, (err) => { this.Toaster(this.serverErrorString[0]);}
```

```
      )}
      //асінхронная функція виводу повідомлень
      async Toaster(m:string){
        const toast = await this.toastCtrl.create({message: m, duration: 2000, position:
'top',  cssClass: 'my-toast-class' }); toast.present();
        }

      // головна функція побудови графіку
      fetchData(datas?: any) {
      Highcharts.setOptions({
          lang: { rangeSelectorZoom: this.selectorZoomString[0],//встановлення масштабу
                      shortMonths: this.shortMonth }        //підключення назв місяців
       });
      // створення графіку
      Highcharts.stockChart('container_chart',{
      credits: {enabled: false}, //підпись highcharts.com  вимкнено
                  chart: {backgroundColor: '#ffffff'},
              rangeSelector: {  //Також форматуємо селектор збільшення графіків.
                  buttons: [{type: 'hour', count: 1, text: this.selectorZoomString[1] },
                           {type: 'hour', count: 3, text: this.selectorZoomString[2] },
                    {type: 'hour', count: 6, text: this.selectorZoomString[3] },
                    {type: 'hour', count: 12,text: this.selectorZoomString[4] },
                    {type: 'day',  count: 3, text: this.selectorZoomString[5] },
                    {type: 'all',            text: this.selectorZoomString[6] }],
                        inputDateFormat:  '%d.%m.%Y',//  Змінюємо  на  звичний  для  нас
формат дати в інтервалах
                  inputEditDateFormat: '%d.%m.%Y',
                  buttonTheme: { width: 80  },// Збільшимо ширину кнопки
                  selected: 5 },    //встановлюємо селектор на потрібну вкладку
                   caption: {text: ''/*Тут може бути текст.*/},
                   title: {  text: ""/*Тут може бути текст.*/},
       //розраховуємо та будуємо вісь X
                   xAxis: {min: Date.now()-(2*24*3600000), max: Date.now()},
      //встановлюємо обмеження осі У
                   yAxis: [{ min: this.item.lmin, max: this.item.lmax}],
                   series: [{
                    type: this.item.linetype,     //встановлюємо тип лінії графіку
                    name: "",                                  //встановлюємо назву лінії
графіку
                    color: this.item.linecolor,   //встановлюємо колір лінії графіку
                    dashStyle:this.item.linestyle,//встановлюємо стиль лінії графіку
                    data:  datas,       //завантажуємо змінну з даними цього графіку
```

```
                        tooltip:{valueDecimals:this.item.decimalsOptions,valueSuffix:" "
+ this.item.suffix,xDateFormat:'%d %b %H:%M:%S'}
                        }] //встановлюємо формат підказки з атрибутами
      });
      }
        //функція вибірки та конвертування останнього параметру архіву в стандартне
значення дати та відповідно обраної мови
       timeConverter(UNIX_timestamp:any){
        let a = new Date(UNIX_timestamp);
        let months = this.shortMonth;
        let year = a.getFullYear();
        let month = months[a.getMonth()];
        let date = a.getDate();
        let hour = "0" +a.getHours();
        let min = "0" +a.getMinutes();
        let sec = "0" +a.getSeconds();
         let time = date + ' ' + month + ' ' + year + ' ' + hour.substr(-2) + ':' +
min.substr(-2) + ':' + sec.substr(-2) ;
        return time;
      }
       //тестова функція для налагодження
      ionViewDidLoad() {       console.log('ionViewDidLoad page-item-detail');    }
      }
```

Файл item-detail.html

```
<ion-header>   <ion-navbar>
  <!--назва параметру -->
    <ion-title>{{ item.name | translate}}</ion-title>
  </ion-navbar> </ion-header>
<ion-content>   <div class="item-detail" padding>
   <!-- останній параметр архіву в стандартне значення дати та відповідно обраної
мови-->
        <div>{{ 'LABEL_TIMES' | translate }}</div>
        <ion-row>           <ion-col>{{last_times}}</ion-col><ion-col>{{last_value}}
{{item.suffix}}</ion-col></ion-row> </div>
   <!-- контейнер для побудови графіку-->
   <div id="container_chart" padding style="height: 400px;margin-top: 50px;"> </div>
</ion-content>
```

Файл control-master.ts

```
import { Component } from '@angular/core';
import { IonicPage, NavController } from 'ionic-angular';
import { Item } from '../../models/item';
import { Items } from '../../providers';

@IonicPage()
@Component({
  selector: 'page-control-master',
  templateUrl: 'control-master.html'
})
export class ControlMasterPage {
  currentControl: Item[]= [];;
  constructor(public navCtrl: NavController, public items: Items) {}
  ionViewDidEnter(){   //завантажується після ініціалізації сторінки
      const my_url = 'index.php?ctrl=0';
      this.currentControl = this.items.query(my_url);// завантаження даних
        }
     /*функція для переходу на сторінку з детальною інформацією про вибраний
елемент.   */
  openItem(item: Item) {
    this.navCtrl.push('ControlDetailPage', {
      item: item
    });
  }
}
```

Файл control-master.html

```
<ion-header>  <ion-navbar><!--назва розділу -->
    <ion-title>{{ 'CTRL_MASTER_TITLE' | translate }}</ion-title>
  </ion-navbar></ion-header>
<ion-content class="ion-padding">
  <ion-list>  <ion-item-sliding *ngFor="let item of currentControl">
      <!--назва параметру -->
      <button ion-item (click)="openItem(item)">
        <ion-avatar item-start><img [src]="item.profilePic"/> </ion-avatar>
        <h2>{{item.name | translate}}</h2>
<ion-col *ngIf="item.status === true"> {{'CTRL_EN' | translate }}</ion-col>
<ion-col *ngIf="item.status === false">{{'CTRL_DIS' | translate }}</ion-col>
 <ion-note item-end *ngIf="item.note">{{item.note}}</ion-note>
      </button>
    </ion-item-sliding>
  </ion-list>
</ion-content>
```

Файл control-detail.ts

```
import { Component } from '@angular/core';
import { IonicPage, NavController, NavParams ,ToastController} from 'ionic-angular';
import { Items, Api} from '../../providers';
import { TranslateService } from '@ngx-translate/core';

@IonicPage()
@Component({  selector: 'page-control-detail', templateUrl: 'control-detail.html'
})
export class ControlDetailPage {
//ініціалізація  змінних
private knobValues = { upper:0, lower:0 };  rMin: any ;  rMax: any ;
serverResponseString:string;controlTitleString:string;enableProccesString:string;
resetButtonString:string;acceptButonString:string; activityProcess: boolean;
rangeEnable: boolean;  btnSaveDisabled: boolean;   item: any;


  constructor(public navCtrl: NavController,
           public navParams: NavParams, public items: Items, public api: Api,
         public toastCtrl: ToastController, private translate: TranslateService
  ) {
  this.item = navParams.get('item') || items.defaultItem;
   this.activityProcess = this.item.status; this.rangeEnable=this.activityProcess;
  this.btnSaveDisabled=true;this.rMin=this.item.lmin;this.rMax=this.item.lmax;
  this.knobValues = { upper:this.item.cmax,   lower:this.item.cmin  }
  //тестова функція для налагодження
  console.log(this.knobValues.upper); console.log(this.knobValues.lower);
   console.log(this.activityProcess);  }
//Функція запускається, коли сторінка повністю введена і тепер є активною сторінкою.
 ngAfterViewInit ( ) { //завантажуємо змінну та статуси поточного параметру
this.translate.get('SERVER_STATUS').subscribe((value) => {
       this.serverResponseString = value;  });
this.translate.get('CTRL_DETAL_PAGE_STRING').subscribe((value) => {
      this.enableProccesString = value[0];  this.resetButtonString = value[1];
      this.acceptButonString = value[2];});
}
  //тестова функція для налагодження
 ionViewDidLoad() {      console.log('ionViewDidLoad page-control-detail');}


enable_date(e) {//функція активація кнопки збереження нових значень параметру
```

```
            this.btnSaveDisabled = false;  this.activityProcess = e._value;
            this.rangeEnable=this.activityProcess
        }
         sSetting(){/*функція створення запиту до серверу та отримання підтвердження збереження нових значень параметру*/
         let msg = 'test'; this.btnSaveDisabled = true;
          const my_url = 'index.php?ctrl='+this.item.nicname+'&stat='+this.activityProcess+'&min='+this.knobValues.lower+'&max='+this.knobValues.upper;
              //створення запиту до серверу
         this.api.get(my_url).subscribe((res:any) => { if (res != null) {
            if (res.status == 'success'){// отримання підтвердження збереження нових значень параметру
           this.Toaster(this.serverResponseString[3]); //вивід повідомлення
            } else {// отримання відхилення запиту
              this.Toaster(this.serverResponseString[2]); this.resetSetting();
            }}else {//отримання відхилення запиту
              this.resetSetting(); this.Toaster(this.serverResponseString[1]);}
          }, err => {//помилка з’єднання
                this.resetSetting(); this.Toaster(this.serverResponseString[0]);
          });       }
            //асінхронная функція виводу повідомлень
         async Toaster(m:string){
         const toast = await this.toastCtrl.create({ message: m, duration: 2000,
               position: 'top',cssClass: 'my-toast-class'}); toast.present();}
         //функція активація кнопки збереження нових значень параметру
         s3Value(event, nome){this.btnSaveDisabled = false;}
         //функція повернення початкових значень параметру цього сеансу
        resetSetting(){this.knobValues ={upper:this.item.cmax,lower:this.item.cmin}
            this.btnSaveDisabled = true;
        }
       }
```

Файл control-detail.html

```
<ion-header>  <ion-navbar><!--назва параметру -->
    <ion-title>{{ item.name | translate}}</ion-title>
  </ion-navbar></ion-header>
<ion-content class="no-scroll">
       <ion-list> <ion-item><!--назва параметру -->
  <div class="item-detail" padding>
    <h2>{{item.name | translate}}</h2>
  </div>
    </ion-item>  <ion-item><!--статус поточного параметру та його функція -->
     <ion-label>{{ enableProccesString }}</ion-label>
```

```
          <ion-toggle [checked]="activityProcess"(ionChange)="enable_date($event)">
          </ion-toggle>
          </ion-item>
       <ion-item> <ion-avatar item-start> <img [src]="item.profilePic"/></ion-avatar>
          <ion-label    style="font-size:    25px;text-align:center;">{{knobValues.lower}}
{{item.suffix}} <--> {{knobValues.upper}}{{item.suffix}}</ion-label>
       </ion-item>
       <ion-item><ion-item ><!--значення поточного параметру та його функція -->
        <ion-range  color="secondary"  dualKnobs="true"  [min]=rMin  [max]=rMax  step="1"
snaps="true"           pin="true"           [(ngModel)]=knobValues           [disabled]="!
rangeEnable"(ionBlur)="s3Value($event, 'slider3')"><!---->
                  <ion-icon      range-left           style="font-size:      12px;"      >
{{item.lmin}}{{item.suffix}}</ion-icon>
                    <ion-icon      range-right      style="font-size:      12px;"      >
{{item.lmax}}{{item.suffix}}</ion-icon>
      </ion-range>
       </ion-item> <!--кнопки  та відповідні функції -->
         <div padding> <ion-row> <ion-col>
         <button   ion-button  (click)="resetSetting()"  style="margin:0px  0  16px;  white-
space:       normal;"color="green"       block               [disabled]="btnSaveDisabled"
>{{ resetButtonString }}
         </button>   </ion-col>   <ion-col>
         <button ion-button (click)="sSetting()" style="margin:0px 0 16px; white-space:
normal; "color="green" block  [disabled]="btnSaveDisabled"  >{{ acceptButonString }}
         </button>   </ion-col>   </ion-row>
        </div>    </ion-item>    </ion-list>  </ion-content>
```

Файл settings.ts

```
import { Component } from '@angular/core';
import { TranslateService } from '@ngx-translate/core';
import { IonicPage, NavController, NavParams } from 'ionic-angular';
import { Settings } from '../../providers';

@IonicPage()
@Component({  selector: 'page-settings',  templateUrl: 'settings.html'
})
export class SettingsPage {
  options: any;         // Наш об’єкт локальних налаштувань
  oldConect :boolean;   // стан попереднього сеансу із сервером
  pathServer:any;       // шлях до сервера

constructor(public navCtrl: NavController, private settings: Settings,
```

```
                public translate: TranslateService) { }


        //функція запису шляху до сервера
        pathVal(e){ this.settings.setValue("option4", e._value); }
        //завантажується після ініціалізації сторінки
        ionViewDidLoad() {
        // функція завантаження даних із пам'яті програми
          this.settings.load().then(() => {
            this.options = this.settings.allSettings;
            this.oldConect  = this.options.option1;
            this.pathServer = this.options.option4;
              });
        }
      //тестова функція для налагодження
        ngOnChanges() {
          console.log('Ng All Changes');
        }
      }
```

Файл settings.html

```
      <ion-header>
        <ion-navbar><!--назва сторинки -->
          <ion-title>{{ 'SETTINGS_TITLE' | translate }}</ion-title>
        </ion-navbar>
      </ion-header>
      <ion-content>
            <ion-item>  <!-- поле стану попереднього сеансу із сервером -->
              <ion-label>{{ 'SETTINGS_OPTION1' | translate }}</ion-label>
               <ion-icon name="thumbs-down" item-end *ngIf="oldConect == false"> </ion-
icon>
               <ion-icon name="thumbs-up"    item-end *ngIf="oldConect == true"> </ion-
icon>
          </ion-item>
          <ion-item>  <!-- поле введення даних шляху до сервера -->
              <ion-label>{{ 'SETTINGS_OPTION4' | translate }}</ion-label>
                            <ion-input   type="text"   text-left   [(ngModel)]=pathServer
(ionBlur)="pathVal($event)"> </ion-input>
          </ion-item>
          <ion-card-content color="primary"> <!-- поле інформаційного тексту -->
            <p>{{ 'SETTINGS_README' | translate }}</p>
          </ion-card-content>
      </ion-content>
```

Файл ua.json


```
{
  "BACK_BUTTON_TEXT": "Назад",
  "NAME": "Ім'я",
  "PASSWORD": "Пароль",
  "EMAIL": "Email",
  "LOGIN": "Вхід",
  "SIGNUP": "Реєстрація",
  "TAB1_TITLE": "Перегляд",
  "TAB2_TITLE": "Регулювання",
  "TAB3_TITLE": "Налаштування",
  "MAP_TITLE": "Мапа",
  "CTRL_MASTER_TITLE":"Регулювання",
  "CTRL_EN":"Активовано",
  "CTRL_DIS":"Вимкнено",
  "SETTINGS_TITLE": "Налаштування",
  "SETTINGS_OPTION1": "Попередний сеанс",
  "SETTINGS_OPTION2": "Логин",
  "SETTINGS_OPTION3": "Пароль",
  "SETTINGS_OPTION4": "Сервер",
  "SETTINGS_PROFILE_BUTTON": "Edit Profile",
  "SETTINGS_PAGE_PROFILE": "Редагування профілю",
  "WELCOME_TITLE": "Ласкаво просимо",
  "WELCOME_INTRO": "Горячев Олександр - розроблено на Ionic Framework  ",
  "LOGIN_TITLE": "Вхід",
    "LOGIN_ERROR":  "Неможливо увійти. Будь ласка перевірте інформацію Вашого облікового запису та спробуйте увійти ще раз.",
  "LOGIN_BUTTON": "Увійти",
  "SIGNUP_TITLE": "Реєстрація",
    "SIGNUP_ERROR":  "Неможливо створити обліковий запис. Будь ласка перевірте інформацію Вашого облікового запису та спробуйте ще раз.",
  "SIGNUP_BUTTON": "Зареєструвати",
  "LIST_MASTER_TITLE": "Перегляд",
  "LABEL_TIMES": "Станом на:",
  "ITEM_CREATE_TITLE": "Новий елемент",
  "ITEM_CREATE_CHOOSE_IMAGE": "Додати зображення",
  "CARDS_TITLE": "Соціальний",
  "SEARCH_TITLE": "Пошук",
  "LOGOUT":"Вихід з облікового запису",
  "SELECTOR_ZOOM" : ["Масштаб","1г","3г","6г","12г","Доба","Усе"],
   "SERVER_STATUS":  ["Збій з'єднання","Сервер на відповідає","Сервер відхилив Ваш запит","Налаштування прийняті"],
```

"SHORTMONTHS" : ["Січ", "Лют", "Бер", "Кві", "Тра", "Чер", "Лип","Сеп", "Вер", "Жов", "Лис", "Гру"],

"CTRL_DETAL_PAGE_STRING":["Активація процесу","Залишити без змін","Прийняти"],
"AIR_TEMPERATURE":"Температура повітря",
"AIR_HUMIDITY":"Вологість повітря",
"WATER_TEMPERATURE":"Температура води",
"GROUND_HUMIDITY":"Вологість землі",
"CARBON_MONOXIDE":"Чадний газ",
"SETTINGS_README":"Налаштування будуть застосовані при наступному вході в систему моніторингу. Закрийте програму і знову увійдіть в систему з новими налаштуваннями"