МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій

«До захисту допущено»
В.о. завідувача кафедри

_____ Світлана ВАЩЕНКО
____ _____ 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА
на здобуття освітнього ступеня магістр

зі спеціальності 122 «Комп'ютерні науки»
освітньо-професійної програми «Інформаційні технології проектування»
на тему: «Мобільний додаток підтримки надання послуг від енергетичних мікромереж»

Здобувача групи ІТ.м-24 Даниленка Романа Сергійовича
(шифр групи) (прізвище, ім'я та по батькові)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Роман Даниленко
(підпис) (Ім'я та ПРІЗВИЩЕ здобувача)

Керівник доцент кафедри ІТ, к.т.н., доцент Володимир НАГОРНИЙ _____
(посада, науковий ступінь, вчене звання, Ім'я та ПРІЗВИЩЕ) (підпис)

Суми 2023

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри ІТ

Світлана ВАЩЕНКО
«_____» _____ 2023 р.

ЗАВДАННЯ

на кваліфікаційну роботу магістра студентові

Даниленку Роману Сергійовичу
(прізвище, ім'я, по батькові)

1 Тема кваліфікаційної роботи Мобільний додаток підтримки надання послуг від енергетичних мікромереж

затверджена наказом по університету від «08» листопада 2023 р. № 1249-VI

2 Термін здачі студентом кваліфікаційної роботи «11» грудня 2023 р.

3 Вхідні дані до кваліфікаційної роботи інформація про локацію мікромереж, інформація про структуру мікромережі, її поточний та прогнозований стан, інформація про прогноз погоди

4 Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) анотація, вступ, аналіз предметної області, постановка задачі та методи дослідження, моделювання мобільного додатку, реалізація мобільного додатку

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових слайдів презентації) актуальність, мета та задачі проекту, інструменти реалізації, контекстна діаграма у нотації IDEF0, діаграма декомпозиції першого рівня у нотації IDEF0, діаграма варіантів використання, діаграма послідовності, діаграма активності, структура локальної бази даних, практична реалізація, демонстрація роботи мобільного додатку, висновки

6. Консультанти випускної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

Дата видачі завдання _____.

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Дослідження предметної області	07.11.23 – 07.11.23	
2	Визначення вимог до додатку	08.11.23 – 09.11.23	
3	Проектування мобільного додатку	10.11.23 – 14.11.23	
4	Побудова моделей даних та методів їх трансформації	15.11.23 – 20.11.23	
5	Побудова екранів взаємодії з користувачем	21.11.23 – 27.11.23	
6	Розробка сценарієв взаємодії користувача з сервісом управляння мікромережами	28.11.23 – 04.12.23	
7	Оформлення пояснювальної записки	05.12.23 – 11.12.23	

Магістрант _____ Роман ДАНИЛЕНКО

Керівник роботи _____ к.т.н., доц. Володимир НАГОРНИЙ

АНОТАЦІЯ

Тема кваліфікаційної роботи магістра «Мобільний додаток підтримки надання послуг від енергетичних мікромереж».

Кваліфікаційна робота складається з вступу, 4 розділів, висновка, списку використаних джерел, та додатків.

Пояснювальна записка кваліфікаційної роботи містить 84 сторінки, 29 рисунки, 15 таблиць, 2 додатки, 32 джерела.

Перший розділі роботи було проведено детальний аналіз предметної області, та досліджені серед мобільних додатків підтримки надання послуг від електромереж.

У другому розділі було конкретизовано мету проекту та визначені задачі реалізації проекту.

Третій розділ присвячено проектуванню мобільного додатку – його структурно-функціональне проектування, моделювання варіантів використання та проектування локальної бази даних.

У четвертому розділі виконується програмна реалізація мобільного додатку: будується архітектура додатку, створюються необхідні структури для локальної бази даних, інтегруються необхідні для роботи бібліотеки, створюються макети всіх необхідних екранів та елементів списків, створюються всі необхідні для додатку моделі даних та прописуються стилі елементів користувацького інтерфейсу.

Результатом кваліфікаційної роботи магістра є мобільний додаток до ОС Android, що надає користувачам інформацію про поточний та прогнозований стан енергетичних мікромереж, що дозволяє отримувати цю інформації у зручний спосіб.

Ключові слова: мобільний додаток, Android, мікромережа.

ЗМІСТ

ВСТУП

Світ сьогодні перебуває на порозі екологічної кризи, яка несе екзистенційну загрозу для людства. Постійне збільшення споживання енергії, підвищення вартості на неї та потреба обмеження використання вуглеводневих видів палива, спонукають до переходу на відновлювальні джерела енергії. Водночас, відзначається швидкий розвиток технічних, ринкових та економічних змін у галузі електроенергетики. Трендом розвинутих країн є впровадження нових підходів до виробництва електроенергії та її розподілу, які спрямовуються на зменшення використання викопних видів палива та задоволенню зростаючого попиту на електроенергію, разом зі зниженням вуглецевого сліду. Прискоренню процесу переходу на відновлювальні джерела енергії слід завдячувати зниженню вартості отримання фотоелектричної сонячної енергії та вартості отримання енергії вітру, за рахунок збільшення потужностей з виробництва засобів генерації. Необхідно відмітити суттєве зростання частки відновлюваної енергії в загальному обсязі виробленої електроенергії.

Суттєвий вплив на достатньо консервативну сферу електроенергетики має цифрова трансформація, яка полягає у впровадженні сучасних передових технологій. Ці технології дозволяють застосовувати автоматизацію там, де раніше використовувалося “ручне керування”. Цифрова трансформація енергетичного сектора являє собою інтеграцію найсучасніших технологій, спрямованих на автоматизацію та покращення ефективності процесів управління енергією.

Змінюється також характер споживання та генерації електроенергії – відбувається перехід від класичного централізованого до більш автономного розподіленого, через впровадження мікромереж. Запровадження відновлюваних джерел енергії у віддалених регіонах та значна кількість споживачів електроенергії з власними генеруючими установками знижує доцільність “ручного режиму” управління мережею. З огляду зазначених трендів, для моніторингу поточного стану роботи та інформування щодо прогнозів стану роботи мікромереж, є необхідність в

розробці програмного продукту, який буде забезпечувати таких сервіс на мобільних пристроях.

Об’єкт дослідження – підтримка надання послуг від енергетичних мікромереж.

Предмет дослідження – мобільний додаток підтримки надання послуг від енергетичних мікромереж.

Мета дослідження – розробити мобільний додаток підтримки надання послуг від енергетичних мікромереж.

Задля досягнення мети проекту, необхідно вирішити наступні задачі:

– дослідити предметну область та визначити ступінь актуальності роботи;
– провести аналіз серед аналогічних мобільних додатків, що використовуються для моніторингу стану мікромереж;
– провести моделювання процесів створення та використання додатку;
– дослідити сучасні архітектурні підходи розробки мобільних додатків для ОС Android та актуальні бібліотеки, які можна використати при розробці;
– спроектувати Android-додаток та обрати архітектуру і технології його розробки;
– програмна реалізація мобільного додатку підтримки надання послуг від енергетичних мікромереж.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Поточне становище та перспективи розвитку моніторингу та керування енергетичними мікромережами.

Одне з нагальних питань сьогодення є дефіциту енергії, підвищення її вартості, обмеженої кількості викопних видів палива та викид CO_2 при його використанні [1]. Для вирішення цих проблем і уникнення кризових ситуацій пропонується використовувати відновлювані джерела енергії та створювати незалежні енергетичні мікромережі, які мають ряд значних переваг. По-перше, невичерпність відновлюваних джерел енергії та відсутність викидів парникових газів позитивно впливають на екологію загалом. Володіння власною енергетичною мікромережею надає можливість повного контролю над витратами електроенергії та реалізації її надлишків у разі наявності таких. Але, ефективність роботи енергетичних мікромереж значною мірою залежить від особливостей місцевого клімату, які можуть достатньо ускладнювати використання, наприклад, енергії вітру та сонця. Від клімату також залежить ефективність управління мікромережами, через збільшення складності прогнозування.

Щодо загальної концепції вирішення проблем енергетики, вона полягає у використанні технологій "розумних мереж" або Smart Grid. Smart Grid – це електрична мережа, яка може інтелектуально інтегрувати дії всіх підключених користувачів для ефективної поставки стійкої, економічної та безпечної електроенергії, динамічно реагуючи на параметри мережі та керуючи ними в режимі "реального часу" [2].

Мережі нового покоління мають стати кібернетичними системами з інтегрованим інтелектуальним управлінням. Ця трансформація може сприяти енергозбереженню, зокрема через зміну парадигми виробництва електроенергії з "централізованого" на "децентралізоване", зміну логіки виробництва та шляхам розподілу електроенергії, а також застосування новітніх програмних рішень для функціональної оптимізації енергосистеми [3].

Під час функціонування кожної енергетичної мережі виникають труднощі та помилки, тому необхідно постійно моніторити її стан. Також важливо впроваджувати системи управління мікромережею для забезпечення балансу між споживанням та виробництвом електричної енергії. Для забезпечення інформаційної підтримки управління енергетичними мікромережами активно використовуються різноманітні інформаційні системи, найпоширеніші є системи підтримки прийняття рішень.

Система підтримки прийняття рішень це інформаційна система, яка використовує обладнання, програмне забезпечення, вхідні дані, базу моделей і роботу менеджера з метою підтримки всіх етапів прийняття рішень у процесі аналітичного моделювання [4]. Основні властивості системи підтримки прийняття рішень:

– інформаційна підтримка особи, яка приймає рішення під час процесу генерування цілей управління;
– пропозиція напрямків та імплементація пошуку інформації з подальшим її аналізом;
– створення, збереження і впровадження формальних моделей, які описують деталі проблемних ситуацій;
– збереження знань про успішно вирішені проблеми та якими методами вони були вирішені.

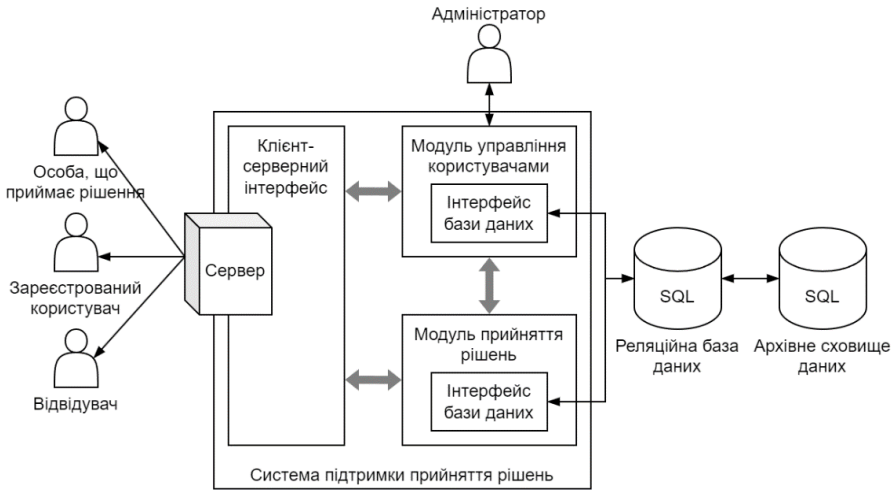

Рисунок 1.1 – Архітектура системи підтримки прийняття рішень з управління мікромережами

Джерело: побудовано автором

При управлінні енергетичними мікромережами архітектура систем підтримки прийняття рішень може бути виконана декількома шляхами. Розглянемо типову архітектуру, що охоплює основні компоненти системи підтримки прийняття рішень, представлену в роботі [5] (рис. 1.1).

Дана системи підтримки прийняття рішень реалізована у вигляді клієнт-серверного додатку. На стороні клієнту надає можливість особі, що приймає рішення відслідковувати стан роботи мережі, аналізувати шляхом моделювання та підключати обчислення процесів з прийняття рішень.

Серверна частина поділена на два модулі: модуль який відповідає за прийняття рішень та модуль який надає користувачам доступ управління. Модуль прийняття рішень керує модулями та екземплярами системи підтримки прийняття рішень, а також операціями, які користувачі можуть виконувати над ними, відповідно до їхніх ролей. Модуль управління користувачами, в свою чергу використовується лише адміністраторами для управління іншими типами користувачів та їх ролей. Вся інформація, пов'язана із системою підтримки прийняття рішень, збережена в базі даних та періодично архівується сховищем даних.

Для прийняття вірних рішень з метою економії матеріальних та часових ресурсів потрібно зібрати та проаналізувати даних, передбачити майбутні події так застосувати необхідні методи підтримки прийняття рішень. Існують два підходи до цього: якісний та кількісний. Якісний підхід ґрунтується на досвіді та інтуїції експертів, а кількісний – на математичних моделях та історичних даних. Кількісний підхід, у свою чергу, ділиться на два типи: причинно-наслідкові моделі та моделі часових рядів [6].

Знаходження простих, адекватних, зручних для застосування на практиці та інформативних методів з прогнозування погодних умов є важливим завданням. Серед них ефективні методи, що базуються на створенні експертних систем, основаних на нечіткій логіці та нейронних мережах. Використання нечіткої логіки дозволяє описати причинно-наслідкові зв'язки між факторами ризику та надати конкретний прогноз чи діагноз у вигляді природної мови, що дозволяє логічно формалізувати експертний

висновок [7]. Для підтримки прийняття рішень також використовується нечітка логіка.

Механізм нечіткого логічного виведення рішення – це процес, який дозволяє приймати рішення на основі нечітких даних. Він здійснюється за допомогою продуційних правил, які складаються експертами. Ці правила представляють собою зв'язки між нечіткими множинами.

Щодо моніторингу даних, широке застосування отримали погодні API. Серед них існують як безкоштовні, так і платні варіанти з різними можливостями, такими як отримання щогодинного або щохвилинного прогнозу. Інформація включає тип опадів, час початку та інтенсивність опадів, температуру, швидкість вітру та індекс УФ-випромінювання. Також важливо враховувати оптимальні погодні умови для конкретних дій, наприклад, для господарювання, включаючи вологість ґрунту. Моніторинг також включає збір даних за допомогою різноманітних датчиків та лічильників.

Розроблена інформаційна система аналізує поточні та прогнозні дані, надає рекомендації особам, відповідальним за прийняття рішення, стосовно структурних змін у мікромережі. Система служить інструментом для вибору ефективного рішення серед доступних альтернативних режимів електромережі, які відповідають технічному стану складових, рівня споживання та прогнозованих погодних умов [8]. Однак дана інформаційна система вимагає доробки, оскільки в ній відсутній інтерфейс для роботи з клієнтами.

1.2 Аналіз існуючих програмних продуктів-аналогів

Сучасні інформаційні технології мають широке застосування в багатьох галузях, зокрема, в енергетиці. Мікромережі з відновлюваними джерелами енергії, які оснащені ІТ-технологіями, є перспективними та універсальними. Вони дозволяють ефективніше використовувати ресурси та підвищувати енергоефективність. Тому мобільні додатки, які взаємодіють з системами підтримки прийняття рішень, що

використовуються для управління енергетичними системами чи моніторингу їх стану, мають важливе значення.

1.2.1. Enphase Enlighten. Enphase Energy.

Програмний продукт Enphase Enlighten від компанії Enphase Energy пропонує широкий спектр функцій, які дозволяють споживачам електроенергії моніторити їхнє споживання електроенергії та вироблення електроенергії з фотоелектричних модулів Enphase [9]. Один з головних екранів додатку показано на рисунку 1.2.

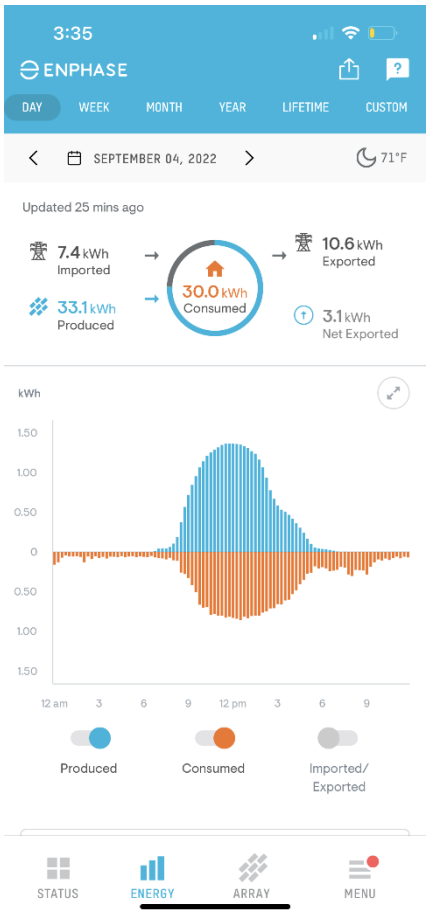

Рисунок 1.2 – Один з екранів додатку Enphase Enlighten

Джерело: [9]

- дозволяє користувачам отримувати інформацію про їхнє поточне споживання електроенергії, вироблення електроенергії, баланс потужностей та інші;
- надає можливість користувачам отримувати інформацію про їхнє споживання електроенергії та вироблення електроенергії за певний період часу, наприклад, за день, тиждень, місяць або рік;
- користувачі можуть отримувати повідомлення про можливі проблеми з мережею, наприклад, про відключення електроенергії або про підвищення вартості електроенергії;
- надає можливість користувачам отримувати аналіз їхнього споживання електроенергії, щоб вони могли виявити способи його зниження;
- дозволяє користувачам керувати своєю системою фотоелектричних модулів Enphase, наприклад, змінювати налаштування або відстежувати роботу модулів.

1.2.2. µGrid Manager. GridWhiz Thailand.

Програмний продукт µGrid Manager від компанії GridWhiz Thailand – це мобільний додаток, який дозволяє операторам керувати їхньою мікромережею [10]. На рисунку 1.3 наведено зображення деяких екранів додатку.

Рисунок 1.3 – Зображення додатку µGrid Manager з Google Play

Джерело: [10]

- дозволяє користувачам отримувати інформацію про поточний стан мікромережі, включаючи споживання електроенергії, виробництво електроенергії та баланс потужностей;
- надає користувачам інформацію про стан їхньої мікромережі за певний період часу, наприклад, за день, тиждень, місяць або рік;
- надсилає повідомлення користувачам про можливі проблеми з їхньою мікромережею, наприклад, про перевантаження мережі або про відключення електроенергії;
- пропонує користувачам широкий спектр функцій з керування пристроями в їхній мікромережі, наприклад, змінювати їх налаштування;
- доступний лише для операторів мікромереж та деякі функції доступні лише в преміум-версії.

1.2.3. mySunPower. SunPower.

Програмний продукт mySunPower від компанії SunPower дозволяє власникам систем сонячної енергії SunPower моніторити їхню систему та керувати нею [11]. Один з головних екранів додатку зображено на рисунку 1.4.

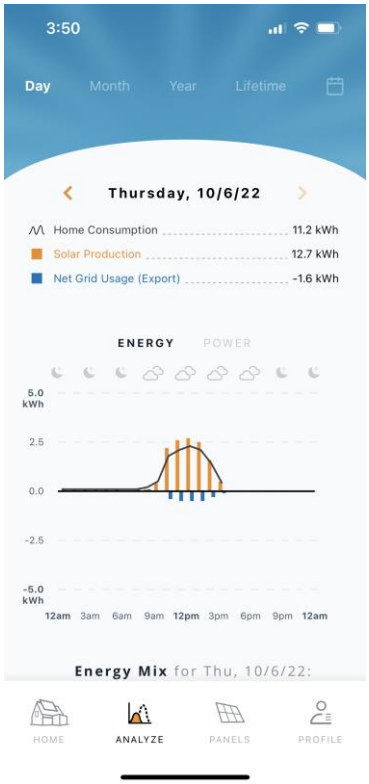

Рисунок 1.4 – Сторінка статистики додатку mySunPower

Джерело: [11]

– пропонує широкий спектр функцій, які дозволяють власникам систем сонячної енергії SunPower моніторити їхню систему та керувати нею;
– простий і інтуїтивно зрозумілий інтерфейс, який дозволяє власникам легко розуміти та використовувати додаток;
– деякі функції, такі як керування системою, доступні лише в преміум-версії.

1.2.4. SMA Energy. SMA Solar Technology AG.

Програмний продукт SMA Energy від компанії SMA Solar Technology AG – це мобільний додаток, який дозволяє споживачам електроенергії, які використовують інвертори SMA, моніторити їхнє споживання електроенергії, а також вироблення електроенергії з фотоелектричних модулів [12]. Додаток пропонує широкий спектр функцій, які дозволяють споживачам електроенергії, які використовують інвертори SMA, моніторити їхнє споживання електроенергії та вироблення електроенергії з фотоелектричних модулів. Основні екрани додатку наведені на рисунку 1.5.

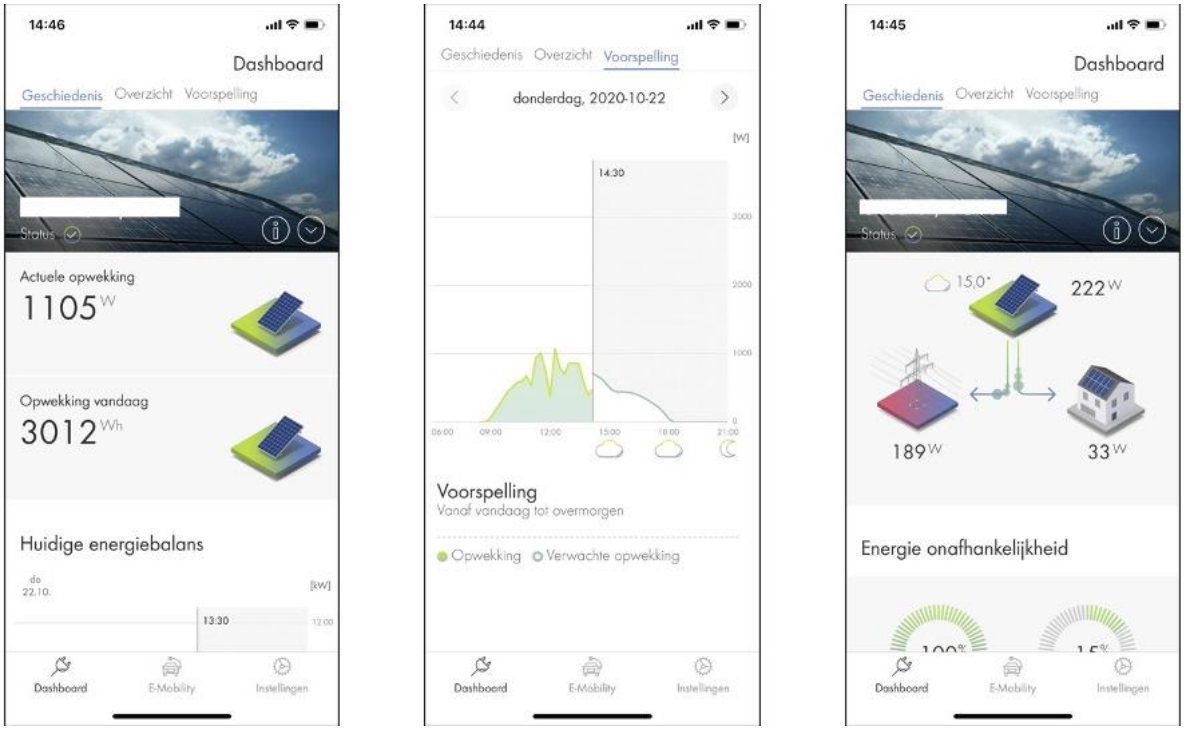

Рисунок 1.5 – Основні екрани додатку SMA Energy

Джерело: [12]

– надає користувачам інформацію про поточний стан їхньої мережі, включаючи споживання електроенергії, вироблення електроенергії, баланс потужностей та інші;
– інформує користувачів про стан їхньої мережі за певний період часу, наприклад, за день, тиждень, місяць або рік;

– надсилає користувачам повідомлення про можливі проблеми з їхньою мережею, наприклад, про відключення електроенергії або про підвищення вартості електроенергії;
– надає доступ до керування своєю системою, наприклад, змінювати налаштування або відстежувати роботу модулів;
– доступний лише для споживачів, які використовують інвертори SMA та деякі функції, такі як керування системою, доступні лише в преміум-версії.

Порівняльні характеристики програмних продуктів наведено у таблиці 1.1.

Аналіз програмних продуктів показав, що всі досліджені додатки надають користувачам інформацію про поточний стан мікромереж та інформацію з історичних даних стану мікромереж. Також варто відмітити, що більшість рішень орієнтуються на роботу з продукцією (чи фотоелектричні модулі, чи інвертори, тощо) власного виробництва, або на ручне керування мікромережами. Окрім цього жодний з досліджених програмних продуктів не надає інформації щодо прогнозованих показників стану мікромереж.

Таблиця 1.1 – Характеристики програмних продуктів

	Enphase Enlighten	µGrid Manager	mySunPower	SMA Energy	Власна розробка
Доступність	Для користувачів фотоелектрич-них модулі Enphase	Для операторів мікромереж	Для власників систем сонячної енергії SunPower	Для користувачі в інверторів SMA	Для визначних користувачів, відповідно до прав доступу
Керування пристроями в мережі	–	+	–	–	–

Продовження таблиці 1.1

	Enphase Enlighten	μGrid Manager	mySunPower	SMA Energy	Власна розробка
Доступ до даних про стан мережі в режимі реального часу	+	+	+	+	+
Доступ до історичних даних про стан мережі	+	+	+	+	–
Прогнозування стану мережі	–	–	–	–	+

2 ПОСТАНОВКА ЗААЧІ ТА МЕТОДИ ДОСЛІДЖЕННЯ

2.1 Мета та задачі дослідження

Мета даної магістерської роботи є надання користувачам енергетичних мікромереж з відновлювальними джерелами енергії доступу до інформації про їх поточний та прогнозований стан.

Задачі дослідження:

– Дослідити предметну область та визначити ступінь актуальності роботи. Встановити стан та перспективи розвитку використання мобільних додатків для роботи з енергетичними мікромережами, сформувати функціональні вимогами до таких мобільних додатків.
– Провести аналіз серед аналогічних мобільних додатків, що використовуються для моніторингу та керування енергетичними мікромережами. Визначити їх цільову аудиторію, провести аналіз досвіду користувача цих додатків та встановити їх функціональні можливості.
– Провести моделювання процесів створення та використання мобільного додатку (представлені в додатку А), що змалює повну картину роботи додатку та розробити детальний поетапний план його програмної реалізації. Структурно-функціональне моделювання, моделювання варіантів використання, деталізація проекту за методом SMART, планування змісту робіт та структури виконавців, розробка діаграми Ганта та планування ризиків проекту.
– Дослідити сучасні архітектурні підходи розробки мобільних додатків для ОС Android та актуальні бібліотеки, які можна використати в розробці. А саме: обрати архітектуру додатку, підібрати необхідні бібліотеки для роботи з web-API та локальною базою даних, а також модуль для шаблону проектування впровадження залежностей.
– Спроектувати Android-додаток згідно з встановлених архітектурних рішень та обраних бібліотек. Спроектувати загальну структуру проекту;

описати необхідні моделі даних для взаємодії з API-сервісами, внутрішні моделі там моделі для зберігання в локальній базі даних; описати сценарії взаємодій процесів мобільного додатку; створити макети користувацького інтерфейсу.

– Програмно реалізувати мобільний додаток, згідно функціональних вимог, та проекту архітектури програмного продукту, що були розроблені. Інтегрувати з інформаційною системою енергетичних мікромереж. Підготовити додаток для розміщення в магазині додатків Play Google.

2.2 Методи дослідження.

Щоб вирішити поставлені задачі використано наступні методи:

– Для вивчення актуальності використання мобільних додатків для отримання доступу до інформації про поточний та прогнозований стан енергетичних мереж застосовуємо аналітичний метод, що включає в себе пошук та аналіз відповідної наукової літератури, дослідження існуючих мобільних додатків, а також встановлення функціональних вимог до мобільного додатку.

– Методи структурно-функціонального моделювання, для представлення системи у вигляді функцій, які пов’язані між собою та функцій, що перетворюють вхідні дані у вихідні.

– Метод проектування для розробки UML-діаграм варіантів використання додатку з визначенням акторів та їх взаємодії з системою. Діаграма, яка надає інформацію про взаємодію між об’єктами системи.

– Технології та методи практичної реалізації мобільного додатку:

  ▪ Мобільний додаток буде побудовано за принципом Clean Architecture [13], за яким код поділяється на шари, кожен з яких має свою власну відповідальність.

- Роботу з API-інтерфейсами реалізовано за допомогою Retrofit2 [14] та OkHttp3 [15], які є добре задокументованими та мають потужний функціонал при використанні.
- Взаємодія з локальною базою даних SQLite реалізована через RoomDB [16], що використовує ефективні алгоритми для доступу до бази даних, а також прості та зрозумілі анотації для опису моделей і взаємодії з базою даних.

3 ПРОЕКТУВАННЯ МОБІЛЬНОГО ДОДАТКУ ПІДТРИМКИ НАДАННЯ ПОСЛУГ ВІД ЕНЕРГЕТИЧНИХ МІКРОМЕРЕЖ

3.1 Структурно-функціональне моделювання

IDEF0 (Integration Definition for Function Modeling) – це методологія функціонального моделювання, яка використовується для опису структури та поведінки систем. IDEF0 використовує діаграми для представлення інформації про систему [17].

Діаграма IDEF0 складається з наступних основних елементів:

– Робоча функція (прямокутник) – це основна операція, яка виконується системою.
– Вхід (стрілки ліворуч) – це дані або ресурси, які необхідні для виконання робочої функції.
– Вихід (стрілки праворуч) – це дані або ресурси, які виробляються робочою функцією.
– Контроль (стрілки зверху) – це умови або обмеження, які впливають на виконання робочої функції.
– Підтримка (стрілки знизу) – це ресурси, які необхідні для виконання робочої функції.

На рисунку 2.1 наведено функціональне моделювання процесу надання інформації з моніторингу мікромереж у нотації IDEF0. На вході надходять дані про доступні користувачу мікромережі, структуру мікромережі, стан компонентів мікромережі та погодні умови. З боку керування – авторизація користувача та доступні для нього мікромережі. Механізмами можна виділити доступність підключення до мережі інтернет та доступ до стороннього сервісу з надання інформації про погоду. На виході отримуємо інформацію про обрану користувачем енергетичну мікромережу її компоненти та їх статус.

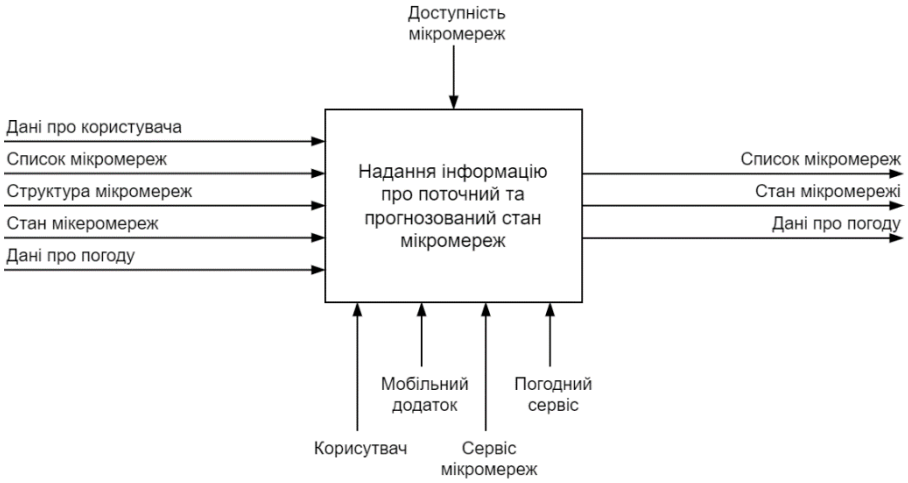

Рисунок 3.1 – Контекстна IDEF0-діаграма додатку

Джерело: побудовано автором

Нотація IDEF0 дозволяє розбивати процеси на більш дрібні складові доти, доки не буде досягнутий необхідний рівень деталізації. Дочірня діаграма описує ту ж саму область, що і батьківська, але з більшою деталізацією. У методології IDEF0 на етапі декомпозиції можуть використовуватися як дочірні, так і батьківськи зв’язки [18]. Наступним кроком є перший рівень декомпозиції діаграми.

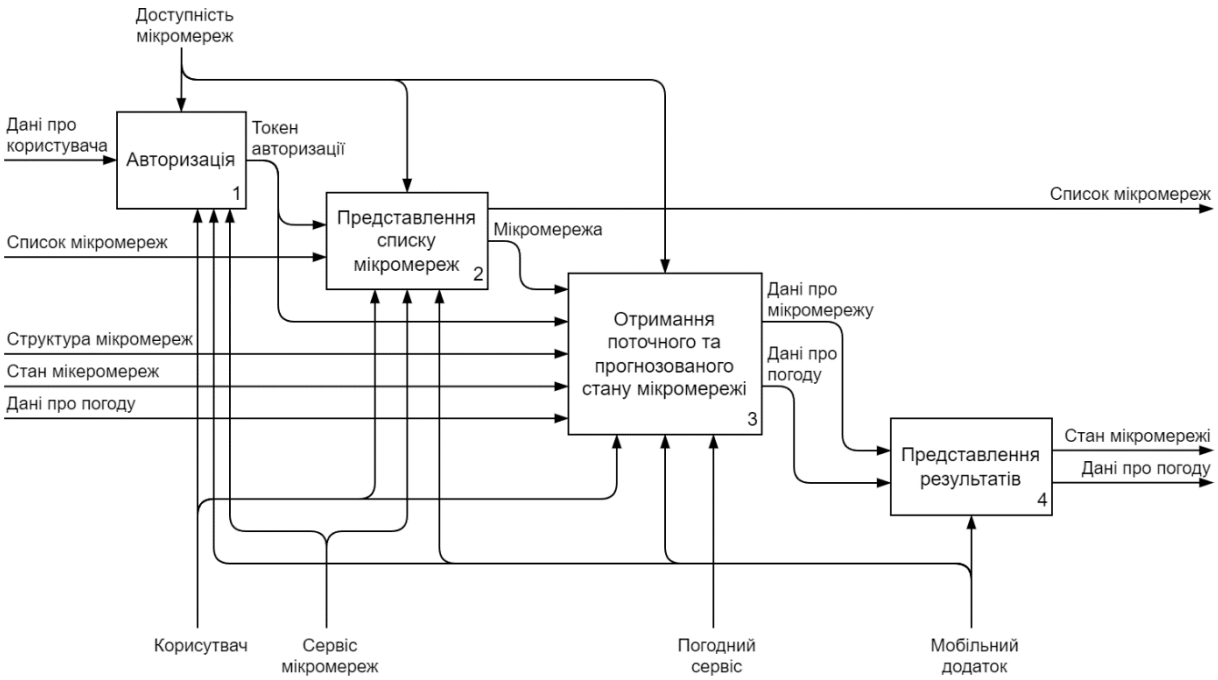

Рисунок 3.2 – IDEF0-діаграма додатку першого рівня

Джерело: побудовано автором

3.2 Моделювання варіантів використання

Use Case діаграма – це опис взаємодій користувача з інформаційною системою [19]. Маємо в системі наступне:

Актори:

- Система моніторингу –частина системи, яка збирає, та зберігає дані про мікромережі та робить прогнози стану мікромереж.
- Зареєстрований користувач – користувач, в якого є доступ до перегляду даних енергетичної мікромережі.
- Сервіс погоди – зовнішній сервіс, який надає інформацію про погоду по заданим координатам.

Варіанти використання:

- Авторизація – дозволяє користувачу авторизуватися в мобільному додатку.
- Перегляд структури мікромережі – надає користувачу дані про структуру мікромережі.
- Перегляд поточного та прогнозованого стану мікромережі – дозволяє користувачу переглядати поточний статус генерації електроенергії та її витрати, а також інформацію щодо накопиченої електроенергії в акумуляторах.
- Перегляд поточної та прогнозованої погоди – дозволяє користувачу переглядати поточну погоду та її прогноз.

Рисунок 2.3 містить діаграму з варіантами використання мобільного додатку з моніторингу мікромереж з відновлюваними джерелами енергії.

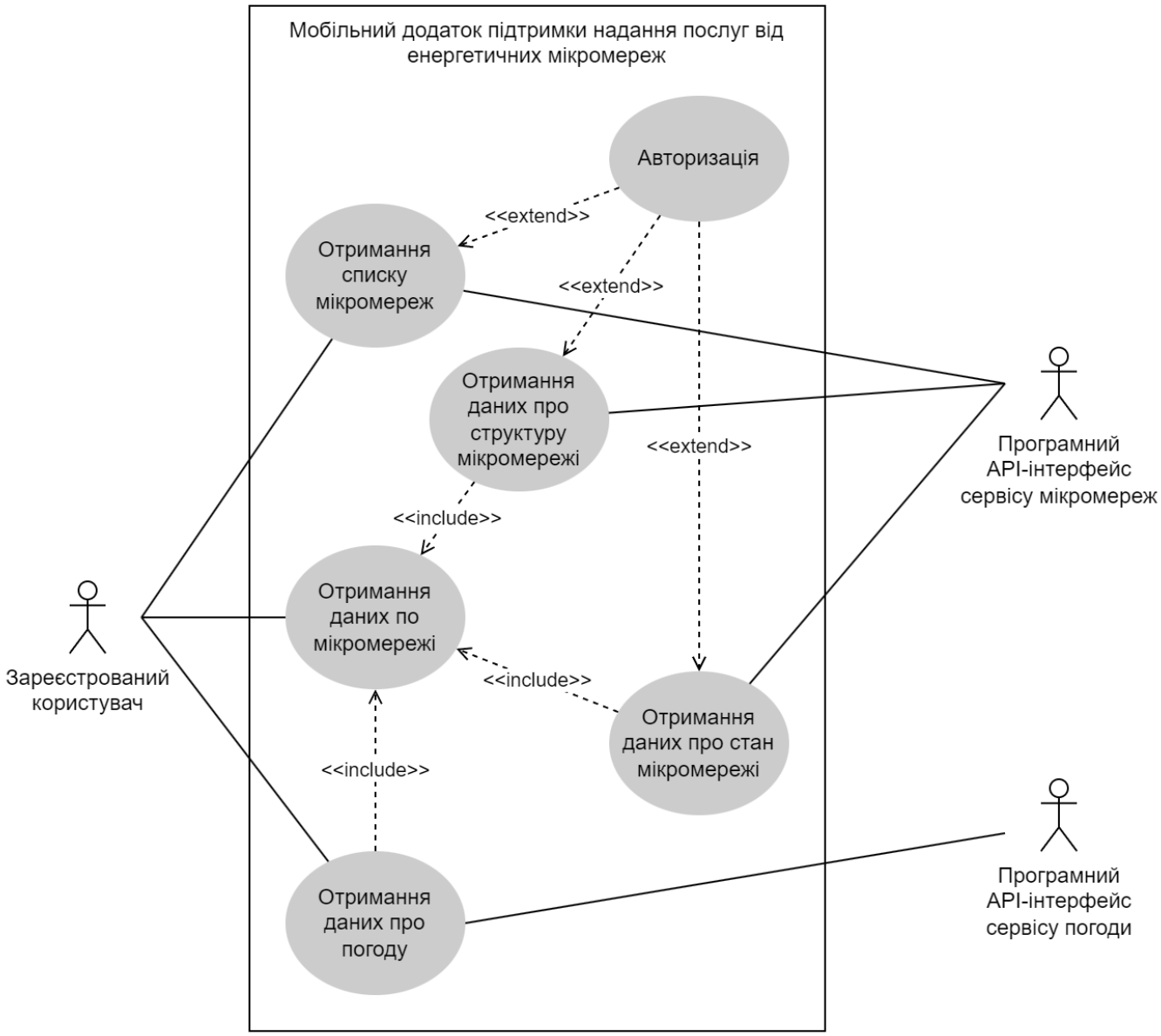

Рисунок 3.3 – Use Case діаграма

Джерело: побудовано автором

Діаграму послідовності мобільного додатку з моніторингу енергетичних мікромереж наведено на рисунку 2.4.

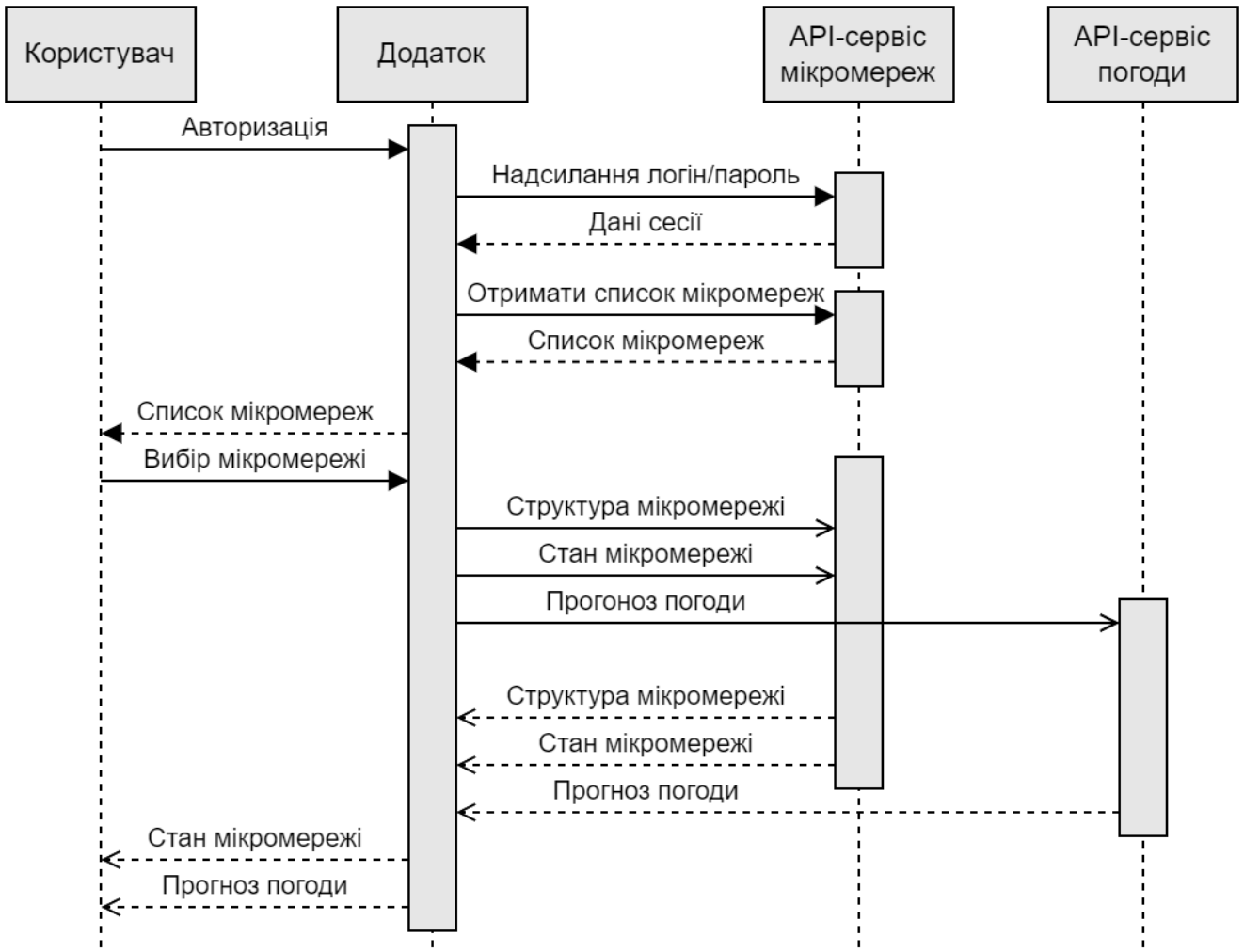

Рисунок 3.4 – Діаграма послідовності

Джерело: побудовано автором

3.3 Проектування бази даних

Під час проектування локальної бази даних було виділено наступні сутності даних: user, location, location_info, generation_item, generation_device weather та weather_hourly. В таблиці 3.1 наведено опис цих таблиць, а на рисунку 3.5 представлена логічна модель бази даних.

Таблиця 3.1 – Таблиці локальної бази даних

Назва таблиці	Опис
user	Дані юзера, що авторизувався
location	Базові дані локацій
location_info	Дані по енергобалансу локації
generation_item	Дані по генеруючим частинам
generation_device	Дані по пристроям генеруючих частин
weather	Дані погоди
weather_hourly	Дані з прогнозу погоди

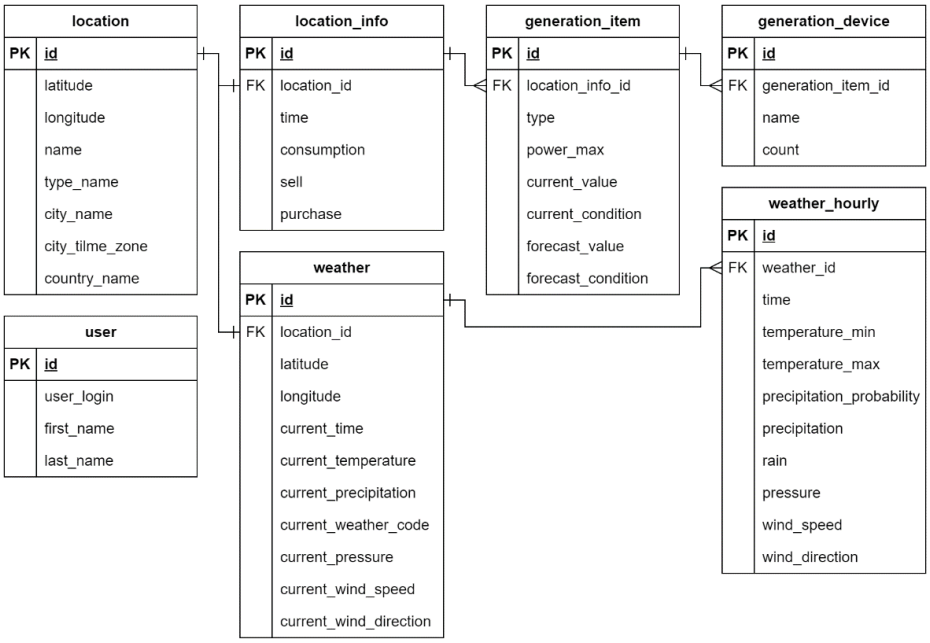

Рисунок 3.5 – Логічна модель бази даних

Джерело: побудовано автором

4 ПРАКТИЧНА РЕАЛІЗАЦІЯ МОБІЛЬНОГО ДОДАТКУ ПІДТРИМКИ НАДАННЯ ПОСЛУГ ВІД ЕНЕРГЕТИЧНИХ МІКРОМЕРЕЖ

4.1 Структура мобільного додатку

При розробці даного мобільного додатку було застосовано архітектуру «Clean Architecture», основною метою якої є розмежування відповідальності між різними частинами додатку (рис. 4.1). Це досягається шляхом розбиття коду на кілька шарів, кожен з яких має свою власну відповідальність і не залежить від інших шарів. Таки підхід дозволяє зробити додаток більш гнучким, масштабованим та зручним для тестування.

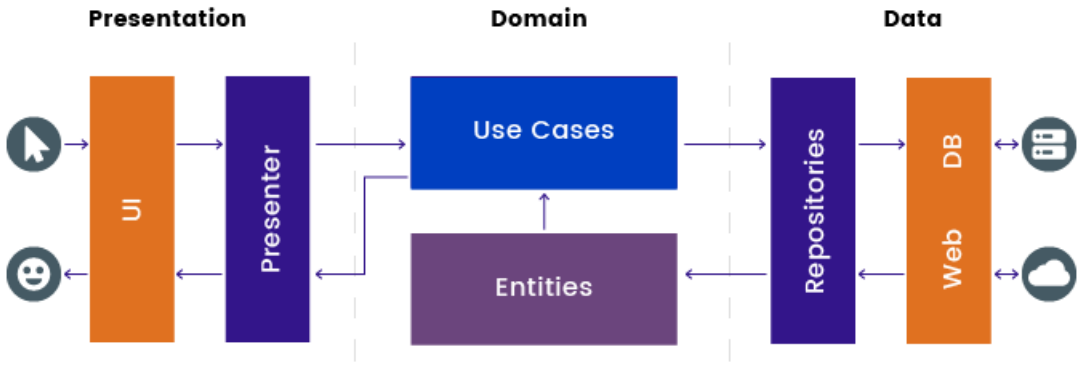

Рисунок 4.1 – Реалізація «Clean Architecture» в Android.

Джерело: [20]

Архітектура передбачає розділення додатку на наступні шари:

– презентація (Presentation) – відповідає за відображення даних так отримання команд від користувача.
– бізнес-логіка (Domain) – відповідає трансформацію даних і має бути незалежний від користувацького інтерфейсу.
– шар даних (Data) – відповідає за доступ до даних як з локальної бази даних, так і з API-інтерфейсів.

Інтерфейс користувача також має свою структуру, яка має слідувати принципам зрозумілості та зручності у використанні. Для нашого мобільного додатку необхідно розробити три екрани, а саме:

– Екран авторизації, на якому користувач буде вводити свій логін та пароль та надсилати запит на авторизацію, для того щоб перейти на наступний екран.
– Головний екран додатку, на якому буде виводитися детальна інформація по вибраній енергетичній мікромережі – її поточний та прогнозований стан роботи, а також інформація про прогноз погоди.
– Екран вибору локацій, на якому користувач зможе перемикатися між своїми мікромережами, для отримання детальної інформації по ним.

Структуру екранів додатку зображено на рисунку 4.2.

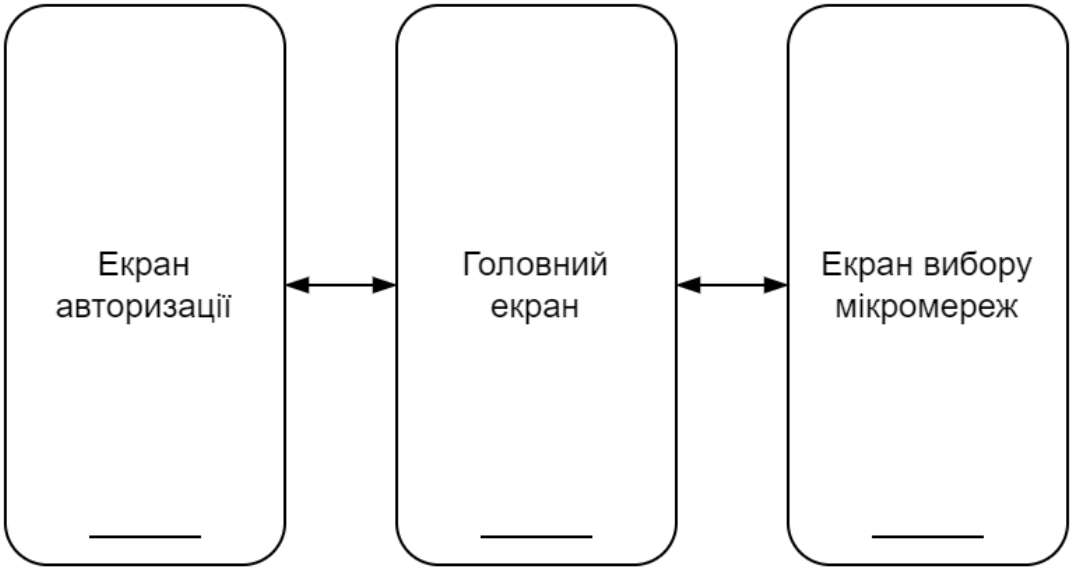

Рисунок 4.2 – Структура екранів додатку

Джерело: побудовано автором

4.2 Реалізація бази даних

Для кожної сутності було визначено необхідні поля та їх тип даних (табл. 4.1 – табл. 4.7).

Таблиця 4.1 – Сутність «user»

Поле	Тип даних	Опис
id	integer	Первинний ключ
user_login	text	Логін користувача
first_name	text	Ім’я користувача
last_name	text	Фамілія користувача

Таблиця 4.2 – Сутність «location»

Поле	Тип даних	Опис
id	integer	Первинний ключ
latitude	real	Широта координат локації
longitude	real	Довгота координат локації
name	text	Назва локації
type_name	text	Тип локації
city_name	text	Назва міста, де розташована мікромережа
city_time_zone	text	Часовий пояс міста
country_name	text	Назва країни

Таблиця 4.3 – Сутність «location_info»

Поле	Тип даних	Опис
id	integer	Первинний ключ
location_id	integer	Зовнішній ключ сутності «location»
consumption	real	Значення споживання енергії
sell	real	Значення продажу енергії
purchase	real	Значення покупки енергії

Таблиця 4.4 – Сутність «generation_item»

Поле	Тип даних	Опис
id	integer	Первинний ключ
location_info_id	integer	Зовнішній ключ сутності «location_info»
type	integer	Тип генеруючої потужності
power_max	real	Максимальна можлива потужність
current_value	real	Поточні значення генерації
current_condition	real	Поточні зовнішні умови
forecast_value	real	Прогнозовані значення генерації
forecast_condition	real	Прогнозовані зовнішні умови

Таблиця 4.5 – Сутність «generation_device»

Поле	Тип даних	Опис
id	integer	Первинний ключ
generation_item_id	integer	Зовнішній ключ сутності «generation_item»
name	text	Назва пристрою
count	integer	Кількість пристроїв

Таблиця 4.6 – Сутність «weather»

Поле	Тип даних	Опис
Id	integer	Первинний ключ
location_id	integer	Зовнішній ключ сутності «location»
Latitude	real	Широта координат
longitude	real	Довгота координат
current_time	integer	Час погодних даних
current_temperatire	real	Поточна температура
current_precipitation	real	Поточні опади
current_weather_code	integer	Поточний код погоди
current_pressure	real	Поточний атмосферний тиск
current_wind_speed	real	Поточна швидкість вітру
current_wind_direction	integer	Поточний напрямок вітру

Таблиця 4.7 – Сутність «weather_hourly»

Поле	Тип даних	Опис
id	integer	Первинний ключ
weather_id	integer	Зовнішній ключ сутності «location»
time	integer	Час прогнозованих даних
temperature_min	real	Максимальна температура
temperature_max	real	Мінімальна температура
precipitation_probability	real	Вірогідність опадів
precipitation	real	Прогноз щодо опади
rain	rain	Прогноз вірогідності дощу
pressure	rain	Прогноз атмосферного тиску
wind_speed	real	Прогноз швидкості вітру
wind_direction	integer	Прогноз напрямку вітру

4.3 Інтеграція бібліотек

«Нативні» додатки для ОС Android в Android Studio можна реалізовувати з використанням мов програмування Java та Kotlin і саме Kotlin є рекомендованою мовою програмування від Google. Kotlin це сучасна мова програмування, яка дозволяє писати компактний та легко читабельний код, має вбудований захист від помилок з null та зручні механізми асинхронного виконання програмного коду.

Для роботи нашого додатку нам необхідна бібліотека для роботи з базами даних SQLite. Було обрано бібліотеку RoomDB, розробником якої є Google і яка має автоматичне управління зв'язками між таблицями, автоматично створює таблиці на основі описів сутностей, виконує всі операції з базою даних у фоновому потоці та має детальну документацію. Архітектуру бібліотеки наведено на рисунку 4.3.

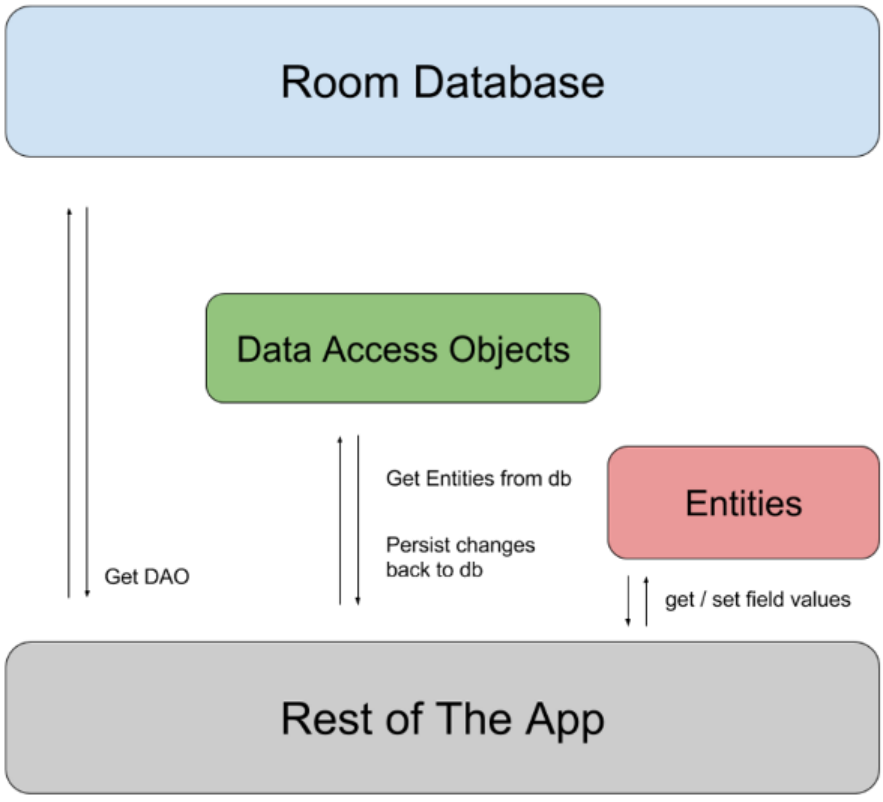

Рисунок 4.3 – Діаграма архітектури бібліотеки RoomDB.

Джерело: [21]

Для взаємодії з REST API при розробці було використано бібліотеку Retrofit2, яка автоматично генерує код для виконання запитів до API на основі інтерфейсів з

анотаціями, має механізми обробки помилок та може виконувати запроси в фонових потоках.

Бібліотека OkHttp3 використовується бібліотекою Retrofit2 і в проекті потрібна для доступу до керування заголовками запитів (наприклад, для додавання даних про авторизацію) та для можливості мати доступ до детальної інформації по запитам і відповідям до web-API під час розробки додатку. На рисунку 4.4 зображено діаграму перехоплення запитів і відповідей, яку надає бібліотека.

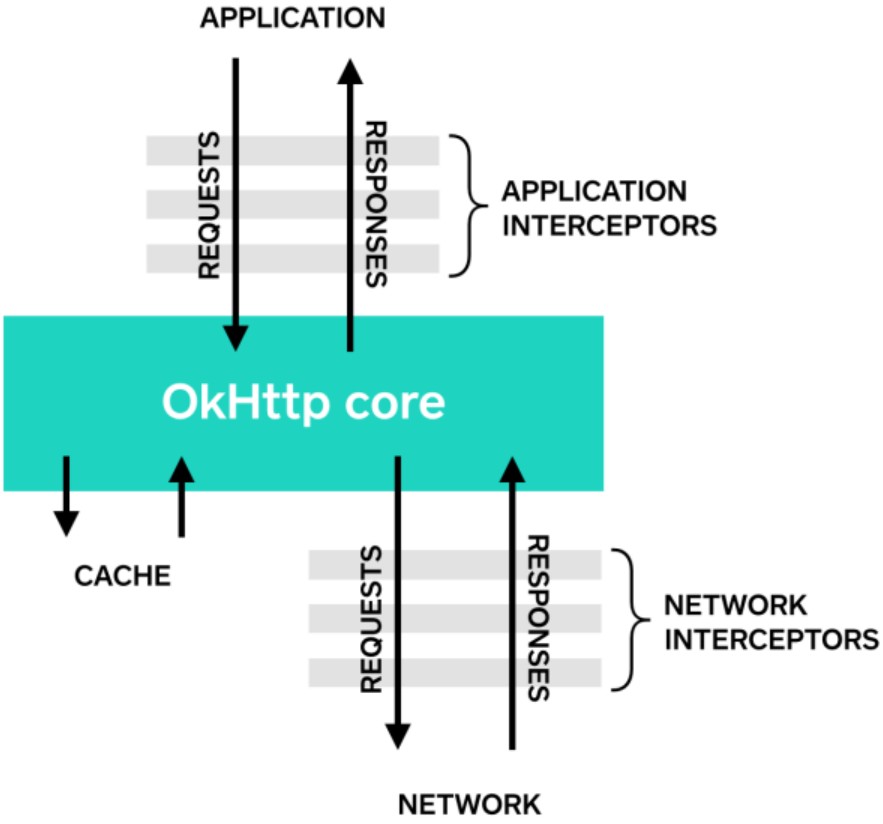

Рисунок 4.4 – Діаграма перехоплення запитів та відповідей у OkHttp.

Джерело: [22]

Для інтеграції шаблону проектування впровадження (або ін’єкції) залежностей (Dependency Injection, DI) використовуємо бібліотеку від розробників мови програмування Kotlin – Koin [23], яка має простий та зрозумілий синтаксис, високу ефективність використання пам’яті та має гнучкі можливості ін’єкцій залежностей.

Для роботи з запитами та відповідями web-API використовуємо формат даних JSON і для зручної конвертації даних між JSON та внутрішніми класами додатку

використовуємо бібліотеку Gson [24] від Google, яка має простий та зручний у використанні синтаксис і що важливо – підтримує різні типи даних при конвертації.

4.4 Реалізація екранів додатку

Перший екран – екран, на якому користувач має можливість ввести свій логін та пароль для авторизації та переходу на наступний екран. У випадку помилки входу на екрані має відображатися помилка, яка виникла при авторизації. Таким чином, на екрані необхідно мати два поля для вводу логіну та паролю, кнопку для початку процесу авторизації. Помилки авторизації будуть відображатися у спливаючих повідомленнях знизу екрану. На рисунку 4.5 наведено макет розташування елементів екрану авторизації.

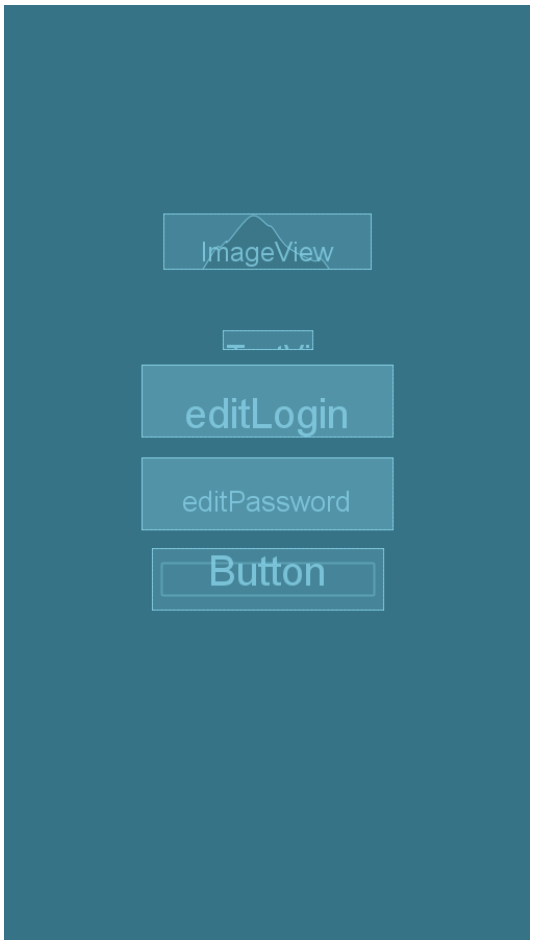

Рисунок 4.5 – Макет екрану авторизації

Джерело: побудовано автором (знімок з екрану)

Наступний екран – є головний екран додатку, на якому відображається інформація про поточний та прогнозований стан роботи обраної енергетичної мікромережі. На екрані маємо наступну інформацію (згори – вниз):

– дату наведених даних по мікромережі;
– кнопка повернення на екран авторизації для зміни користувача;
– дані по споживанню, продажу та покупці електроенергії;
– блок з детальною інформацією по кожній з частин структури мікромережі, які представлені у вигляді горизонтального списку;
– інформація про поточний стан погодних умов на локації;
– кнопка з назвою локації, натиснувши на яку користувач переходить до діалогового вікна зі списком доступних локацій;
– блок з прогнозом погоди на найближчі 4 дні.

Макет екрану авторизації наведено на рисунку 4.6.

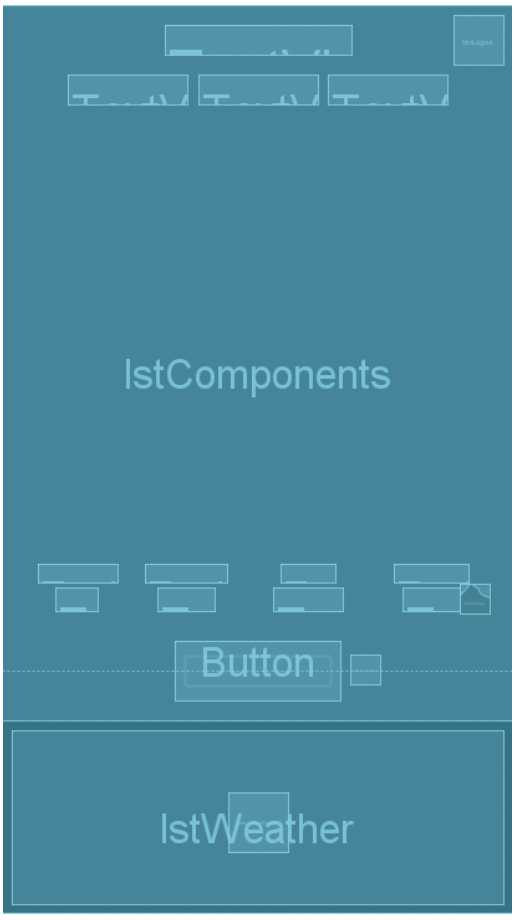

Рисунок 4.6 – Макет головного екрану

Джерело: побудовано автором (знімок з екрану)

Користувач може мати доступ до більше ніж однієї мікромережі, тому необхідно мати екран для вибору локації з мікромережею для детальної інформації по ній. Цей екран реалізовано у вигляді спливаючого діалогового вікна та являє собою список з переліком локацій, кожен елемент списку якого містить в собі назву локації, назву типу локації та її адресу. Макет екрану зображено на рисунку 4.7.

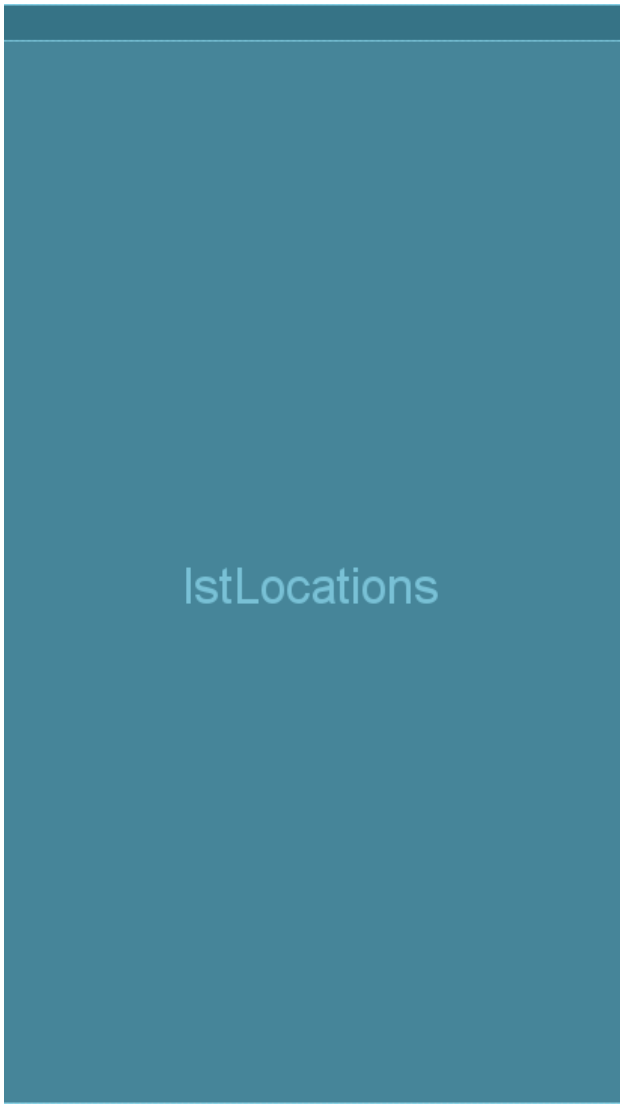

Рисунок 4.7 – Макет екрану вибору локації зі списку доступних локацій

Джерело: побудовано автором (знімок з екрану)

Основним елементом головного екрану є структурні компоненти енергетичної мікромережі. Для кожного типу компоненту передбачено свій власний макет відображення даних. Макети для кожного з компонентів мікромережі наведено на рисунках 4.8 – 4.10.

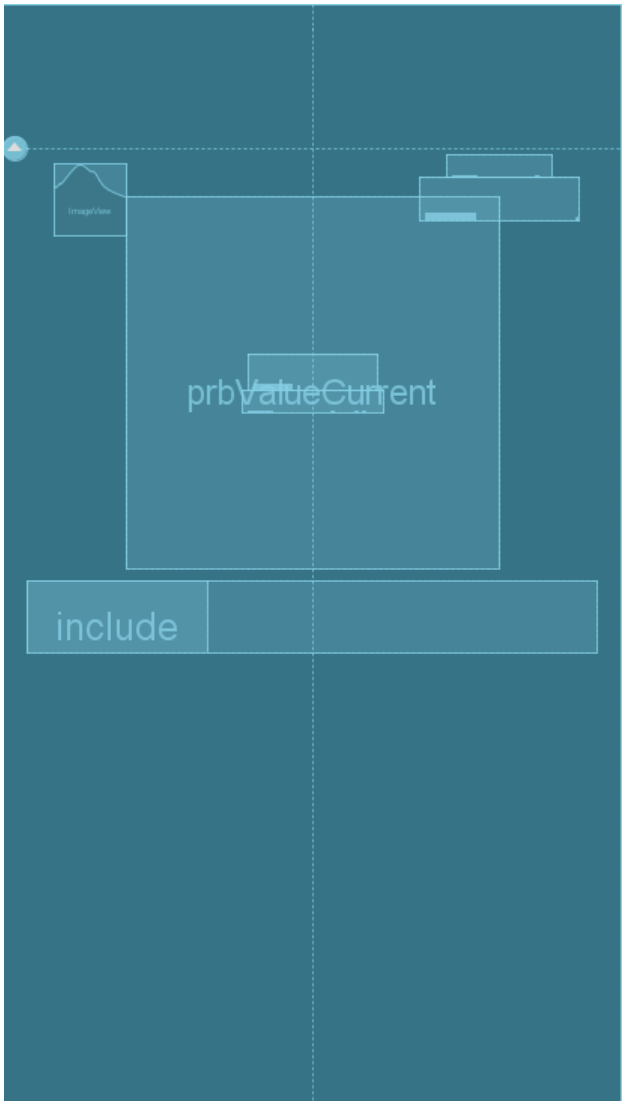

Рисунок 4.8 – Макет елементу списку для компонента енергетичної мікромережі з акумуляторами

Джерело: побудовано автором (знімок з екрану)

Для компонента з акумуляторами передбачено наступні дані для виводу:

– максимальний запас накопичення енергії;
– поточний запас накопиченої енергії у вигляді фактичних значень та у вигляді графічного зображення частки накопиченої енергії;
– список пристроїв які використовуються для накопичення енергії з їх назвою та їх кількістю.

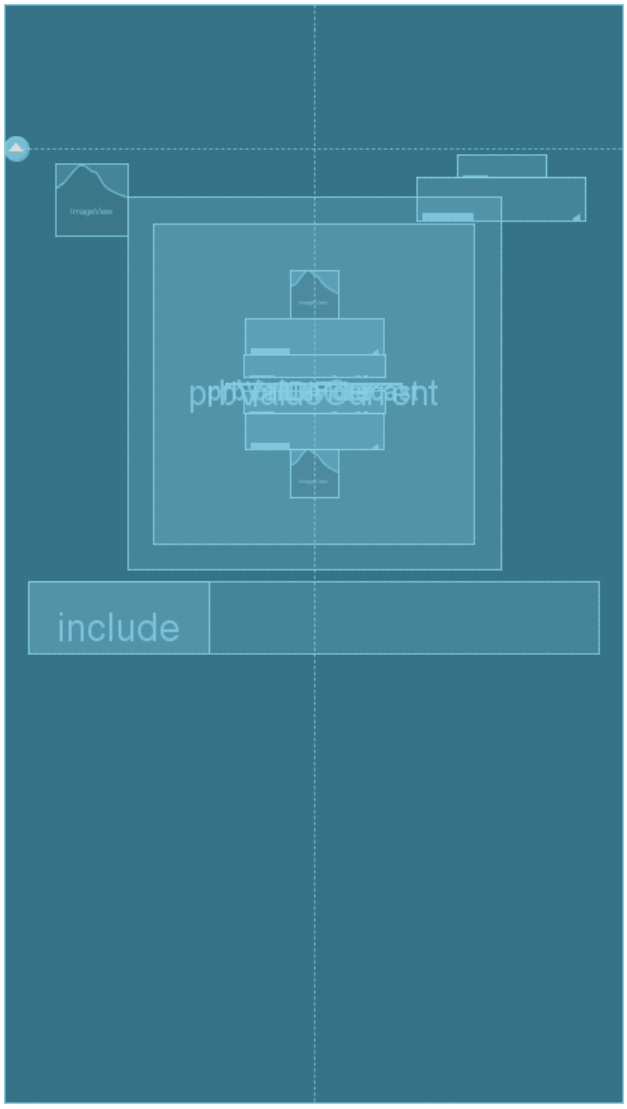

Рисунок 4.9 – Макет елементи списку для компонента енергетичної мікромережі з сонячними панелями

Джерело: побудовано автором (знімок з екрану)

Для компонента з сонячними панелями передбачено наступні дані для виводу:

– максимальна генеруюча потужність
– поточна та прогнозована потужність у вигляді фактичних значень та у вигляді графічного зображення частки генеруючою потужності;
– список пристроїв які використовуються для генерації електроенергії з їх назвою та їх кількістю.

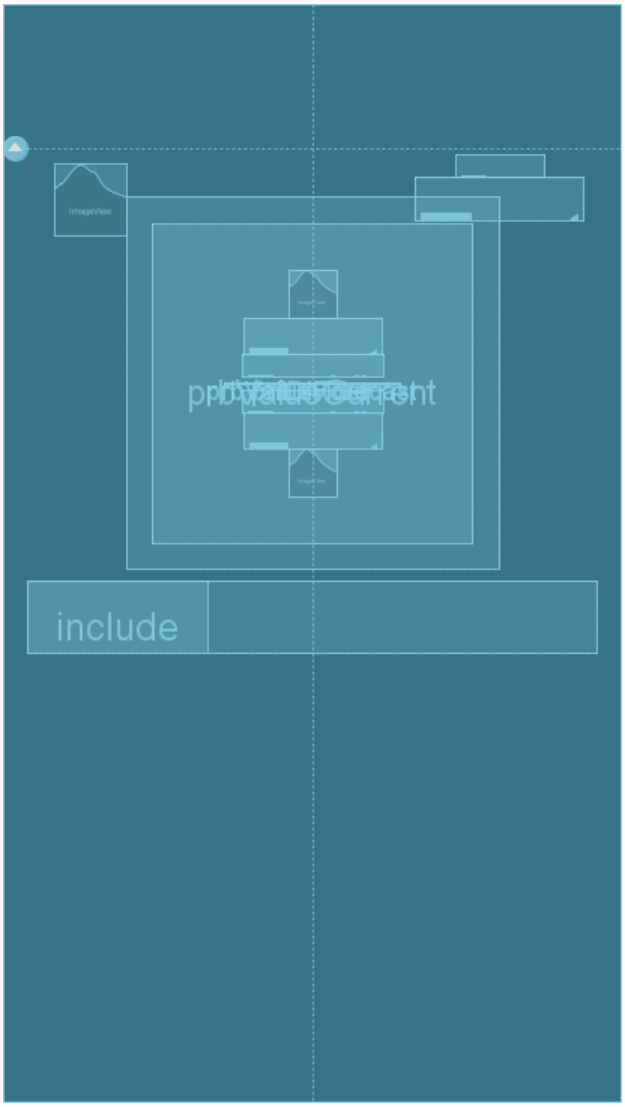

Рисунок 4.10 – Макет елементу списку для компонента енергетичної мікромережі з вітровими турбінами

Джерело: побудовано автором (знімок з екрану)

Для компонента з вітровими турбінами передбачено наступні дані для виводу:

– максимальна генеруюча потужність
– поточна та прогнозована потужність у вигляді фактичних значень та у вигляді графічного зображення частки генеруючою потужності;
– список пристроїв які використовуються для генерації електроенергії з їх назвою та їх кількістю.

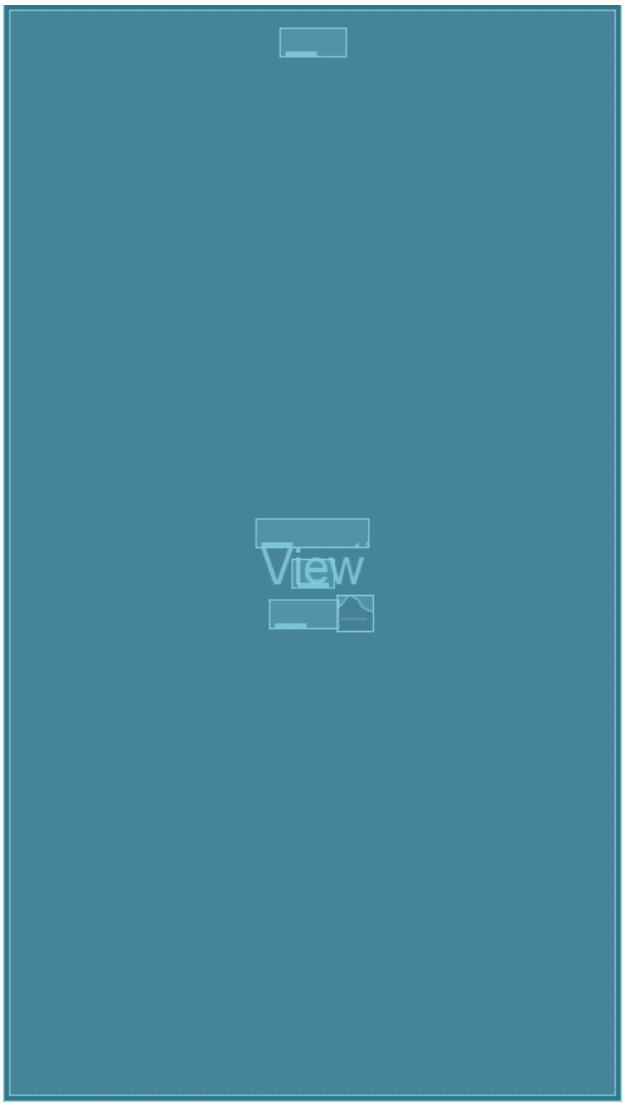

Рисунок 4.11 – Макет елементу списку з прогнозу погоди

Джерело: побудовано автором (знімок з екрану)

Для елементу списку прогнозу погоду маємо окремий макет який наведено на рисунку 4.11, та який має наступні дані для виводу:

– дата на яку наведено прогноз погоди;
– прогноз мінімальної та максимальної температури;
– прогноз вірогідності опадів;
– прогнозована швидкість вітру та його напрямок.

В результаті макетування екранів, отримуємо наведену на рисунку 4.12 структуру макетів сторінок так їх елементів в проекті.

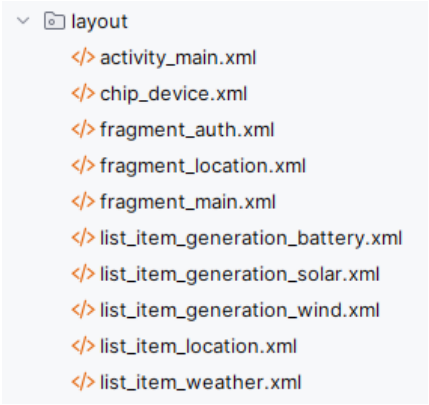

Рисунок 4.12 – Список макетів екранів і їх елементів в проекті

Джерело: побудовано автором (знімок з екрану)

4.5 Реалізація моделей даних

В додатку класи даних розділені на три групи: моделі даних для отримання даних з web-API (DTO-моделі), моделі даних для зберігання даних в локальній базі даних (Entity-моделі) та моделі даних для виводу інформації користувачу (Domain-моделі).

Приклад класу даних для отримання даних по локації з web-API наведено нижче.

```
data class LocationDto(
    @SerializedName("loc_id")
    val locationId: Long,
    @SerializedName("loc_latitude")
    val latitude: Float,
    @SerializedName("loc_longitude")
    val longitude: Float,
    @SerializedName("loc_name")
    val name: String,
    @SerializedName("lt_type_title")
    val typeName: String,
    @SerializedName("city_name")
    val cityName: String,
    @SerializedName("city_time_zone")
    val cityTimeZone: Int,
    @SerializedName("country_name")
    val countryName: String
)
```

Нижче наведено приклад класу даних по локації для зберігання в локальній базі даних.

```
@Entity(tableName = "location")
data class LocationEntity(
    @PrimaryKey
    val id: Long,
    val latitude: Float,
    val longitude: Float,
    val name: String,
    @ColumnInfo(name = "type_name")
    val typeName: String,
    @Embedded(prefix = "city_")
    val city: LocationCityEntity,
    @Embedded(prefix = "country_")
    val country: LocationCountryEntity
)
```

Приклад класу даних по локації для виводу інформації користувачу наведено на нижче.

```
data class Location(
    val id: Long,
    val coordinates: Coordinates,
    val name: String,
    val typeName: String,
    val city: LocationCity,
    val country: LocationCountry,
) : Serializable
```

В Kotlin прийнято використовувати принцип незмінюваності даних, це принцип згідно з яким значення об’єкту класу даних не можна змінювати після його створення. Це дозволяє бути впевненим, що якщо будь який фрагмент коду отримує об’єкт такого класу, його значення не зміняться. Такі класи називаються дата класи (data class), якими є всі класи даних у проекті.

Для перетворення одних типів даних в інші використовуються функції розширення (extension functions). Це такі функції, які розширюють функціональність існуючих класів, але вони не є функціями цих класів, а лише додаються до класів, як додаткові можливості. Такий підхід дозволяє залишати класи даних «чистими» від зайвого функціоналу. Приклад таких функцій наведено нижче.

```
fun LocationDto.toDomain() : Location {
    return Location(
        id = this.locationId,
        coordinates = Coordinates(this.latitude, this.longitude),
        name = this.name,
        typeName = this.typeName,
        city = LocationCity(this.cityName, this.cityTimeZone),
        country = LocationCountry(this.countryName)
    )
}
fun Location.toDao() : LocationEntity {
    return LocationEntity(
        id = this.id,
        longitude = this.coordinates.longitude,
        latitude = this.coordinates.latitude,
        name = this.name,
        typeName = this.typeName,
        city = LocationCityEntity(
            name = this.city.name,
            timeZone = this.city.timeZone
        ),
        country = LocationCountryEntity(this.country.name)
    )
}
```

4.6 Побудова стилів елементів інтерфейсу

Для зручності розробки інтерфейсу додатків для ОС Android прийнято використовувати стилі для елементів інтерфейсу. Таким чином не потрібно для кожного елементу інтерфейсу прописувати всі необхідні параметри які впливають на його вигляд та розташування на екрані. Такі стилі описуються у файлі themes.xml. стилі мають можливість наслідування, що позбавляє необхідності повторювати у стилях до різних елементів однієї групи спільні характеристики. Нижче наведено приклад опису стилів елементів інтерфейсу користувача та наслідування стилів.

```
<style name="Label.Date" parent="Label">
    <item name="android:layout_width">wrap_content</item>
    <item name="android:layout_height">wrap_content</item>
    <item name="android:layout_margin">@dimen/margin_x4</item>
    <item name="android:textSize">@dimen/text_large</item>
    <item name="android:text">@string/date_time_normal_default</item>
</style>
```

```
<style name="Label.Date.Forecast" parent="Label.Date">
    <item name="android:layout_marginTop">@dimen/margin_x4</item>
    <item name="android:layout_marginBottom">@dimen/margin_x2</item>
    <item name="android:textSize">@dimen/text_normal</item>
</style>
```

Після опису всіх необхідних стилів ми отримали завершений вигляд всіх екранів мобільного додатку. Приклад готового дизайну наведено на рисунку 4.13, на прикладі головного екрану.

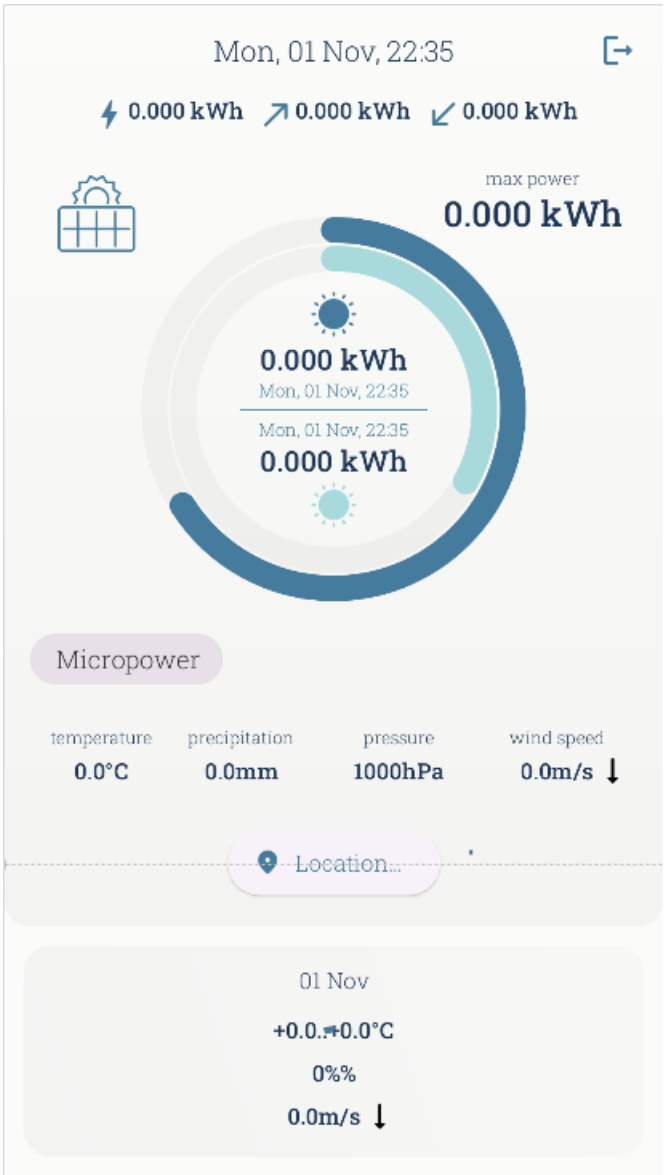

Рисунок 4.13 – Макет головного екрану з застосованими стилями

Джерело: побудовано автором (знімок з екрану)

4.7 Приклад роботи мобільного додатку

На рисунках 4.14 – 4.16 наведено приклад роботи готового мобільного додатку підтримки надання послуг від енергетичних мікромереж.

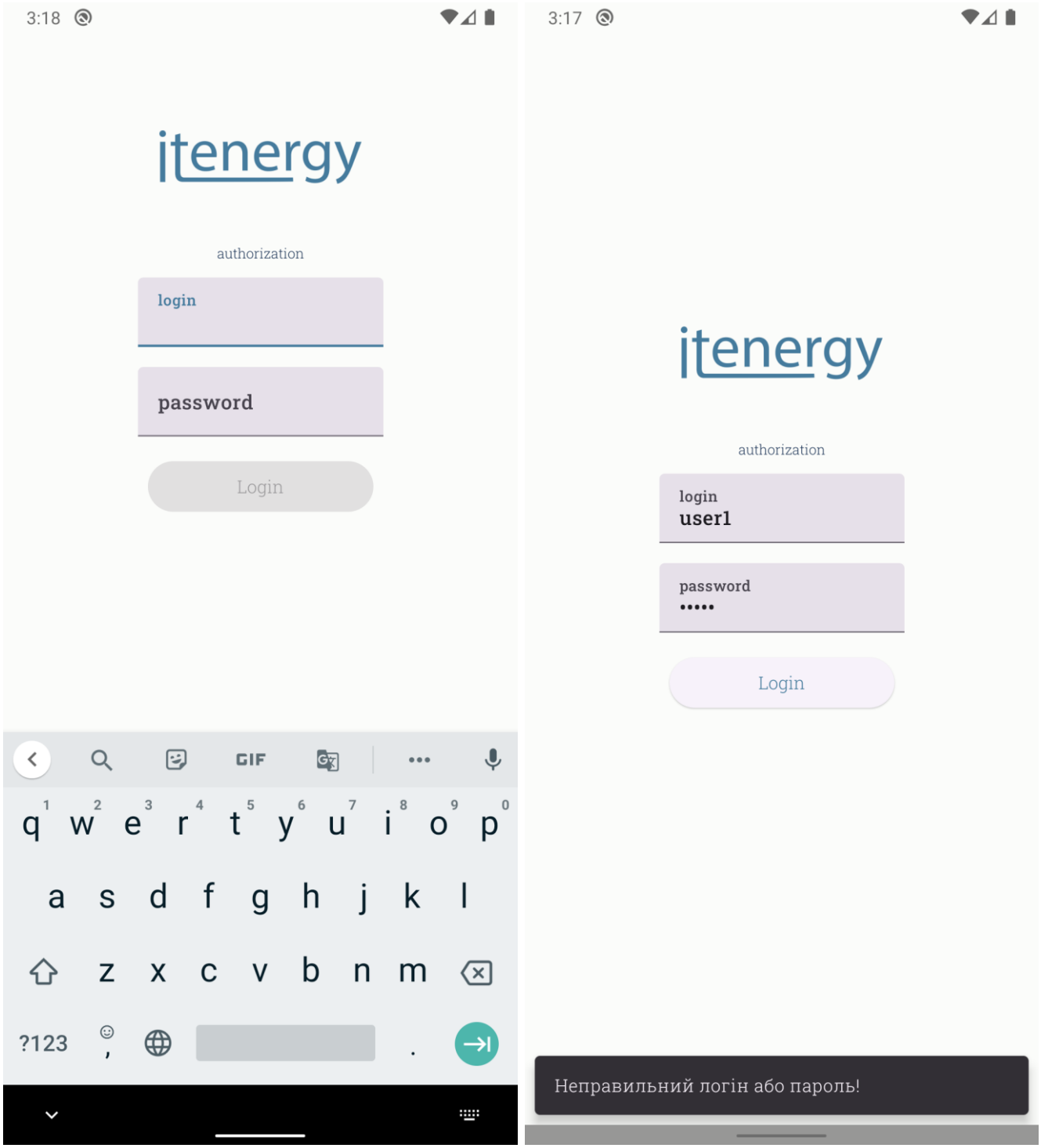

Рисунок 4.14 – Екрани авторизації з пустими полями для воду даних користувача та з повідомленням про помилку авторизації

Джерело: побудовано автором (знімок з екрану)

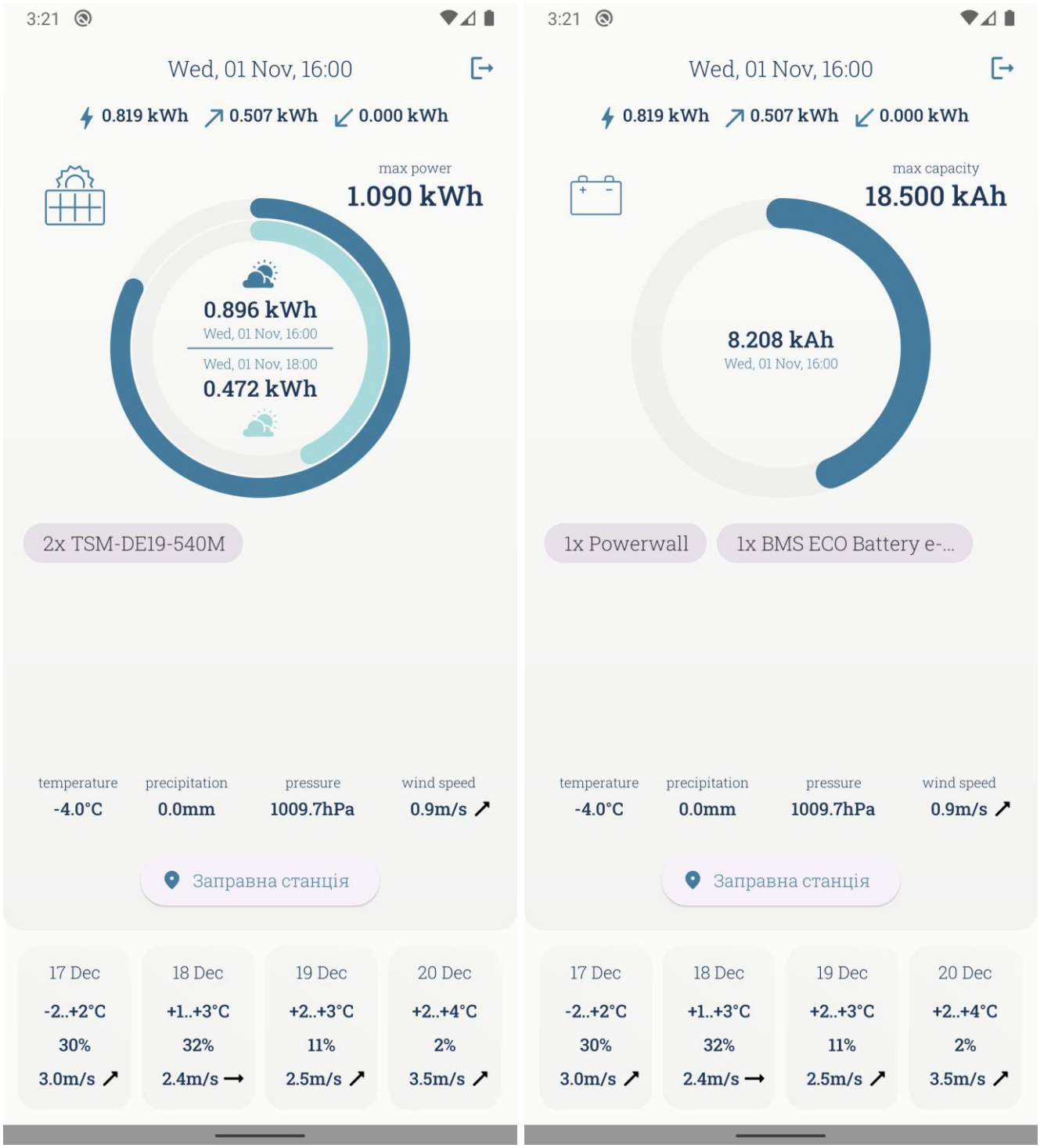

Рисунок 4.15 – Головний екран з інформацією про стан мікромережі та даними по сонячним панелям і накопичувачам електроенергії

Джерело: побудовано автором (знімок з екрану)

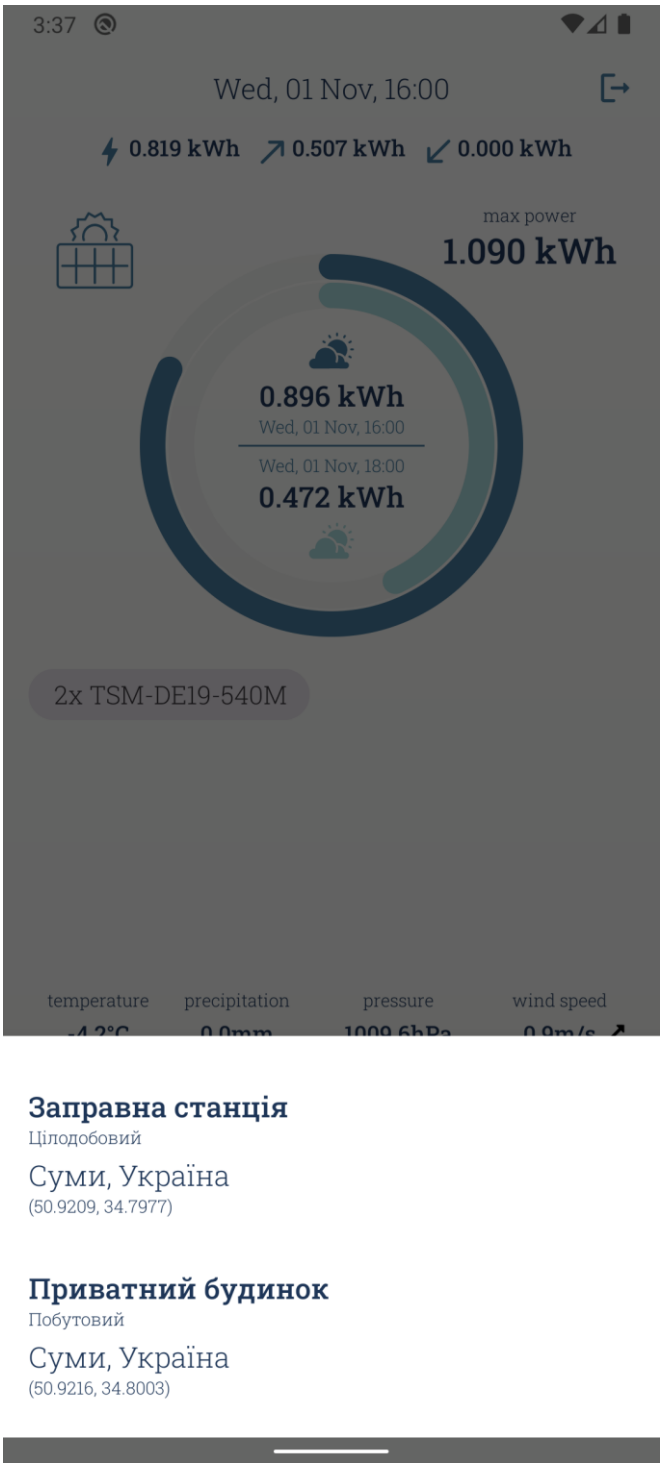

Рисунок 4.16 – Діалог зі списком локацій доступних для вибору

Джерело: побудовано автором (знімок з екрану)

ВИСНОВКИ

В рамках магістерської роботи було проведено дослідження сучасного стану та перспектив трансформації енергетичних мікромереж. В результаті було встановлено, що вони є перспективним напрямком в енергетичній галузі, оскільки подібні мережі підвищують надійність електропостачання, впливають на зменшення викидів парникових газів і збільшують ефективність використання відновлювальних джерел енергії. Роль енергетичних мікромереж під час бурхливого збільшення використання відновлювальних джерел та актуальних екологічних проблем, отримує потужний поштовх і має великі перспективи подальшого якісного розвитку із залученням сучасних технологій, в тому числі залученням інформаційних систем.

З урахування сучасних трендів у сфері розробки мобільних додатків для операційної системи Android, було сформовано технічні та функціональні вимоги до мобільного додатку та проведено планування робіт з його розробки, які дозволять отримати програмний продукт, що буде відповідати актуальним умовам магазину додатків Play Google. Разом з цим були розглянуті підходи та вимоги до проектування мобільних додатків, обрана сучасна гнучка архітектура побудови додатку та визначені необхідні допоміжні бібліотеки. Це все дозволить швидко та ефективно реалізувати потрібні функції програмного продукту і необхідний досвід використання. Під час роботи було проведено структурно-функціональне моделювання процесу використання та розробки мобільного додатку з моніторингу мікромереж.

На заключному етапом магістерської роботи було розроблено мобільний додаток підтримки надання послуг від енергетичних мікромереж. Під час програмної реалізації проекту були використані: мова програмування Kotlin; застосовані бібліотека Koin для реалізації патерну ін’єкції залежностей, бібліотеки Retrofit2 і OkHttp3 для взаємодії з web-API та бібліотеку Gson для перетворення JSON даних; застосовані актуальні підходи до побудови архітектури та досвіду користувача додатків для ОС Android.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Eugene D. Coyle and Richard A. Simmons. Understanding the Global Energy Crisis. Purdue University Press. 2014. 304c.
2. Smart Grids and Renewables: A Guide for Effective Deployment. URL: https://www.irena.org/publications/2013/Nov/Smart-Grids-and-Renewables-A-Guide-for-Effective-Deployment (дата звернення 07.11.2023)
3. Shendryk S., Tymchuk S., Shendryk V., Telizhenko O. Electricity power consumption management in hybrid power grid with renewable energy sources. Information systems and innovative technologies in project and program management: Collective monograph edited by Linde I., Chumachenko I., Timofeyev V. ХНУРЕ, 2019. С. 161-169
4. СППР-систем. *Softline.* URL: https://softline.org.ua/sppr.html (дата звернення 07.11.2023)
5. Bellini P., Pantaleo G., Nesi P. A Smart Decision Support System for Smart City : Conference Paper, December 2015
6. Періодизація майбутнього, методи і засоби його передбачення. *Архів для студентів.* URL: https://studfile.net/preview/16468747 (дата звернення 07.11.2023)
7. Сергета І., Молчанова О. Прогнозування на основі нечіткої логіки і нейронних мереж та основні напрямки його використання у практиці гігієнічних досліджень. Вінницький національний медичний університет ім. М.І. Пирогова, Вінниця
8. Shendryk S., Shendryk V., Parfenenko Y, Drozdenko O., Tymchuk S. Decision Support System for Efficient Energy Management of MicroGrid with Renewable Energy Sources. Proceedings of the 11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2021. 2021. 1. С. 225-230. DOI: 10.1109/IDAACS53288.2021.9660966.

9. Enlighten – the monitoring experience for the system owner. *Play Google.* URL: https://play.google.com/store/apps/details?id=com.enphaseenergy.myenlighten (дата звернення 07.11.2023)
10. The µGrid Manager or The Microgrid Manager *Play Google.* URL: https://play.google.com/store/apps/details?id=th.co.gridwhiz.MicrogridManager (дата звернення 07.11.2023)
11. mySunPower. SunPower. *Play Google.* URL: https://play.google.com/store/apps/details?id=com.mysunpower (дата звернення 07.11.2023)
12. SMA Energy. SMA Solar Technology AG. *Play Google.* URL: https://play.google.com/store/apps/details?id=de.sma.energy (дата звернення 07.11.2023)
13. Dumbravan A., Price E. Clean Android Architecture: Take a layered approach to writing clean, testable, and decoupled Android applications. Packt Publishing Ltd., 2022. 350c.
14. Using Retrofit 2 for web-services in Android with a simple demo project. *Medium.* URL: https://medium.com/cr8resume/make-your-hand-dirty-with-retrofit-2-a-type-safe-http-client-for-android-and-java-c546f88b3a51 (дата звернення 07.11.2023)
15. A Guide to OkHttp. *Baeldung.* URL: https://www.baeldung.com/guide-to-okhttp (дата звернення 07.11.2023)
16. How to use Koin DI with Room Database in Kotlin. Part 1. *Androidgeek.* URL: https://androidgeek.co/part1-how-to-use-koin-di-with-room-database-in-kotlin-436023ac5572 (дата доступу 07.11.2023)
17. Mora M. Impacts of IDEF0-Based Models on the Usefulness, Learning, and Value Metrics of Scrum and XP Project Management Guides // Engineering Management Journal. 2022. Vol. 34, Issue 4. C. 574-590
18. Baghbani M. IDEF0 Modeling Standard: a tool for process map drawing under requirements of ISO 9001: 2015: a case study // Journal of Modern Processes in Manufacturing and Production. Autumn 2019. Vol. 8, Issue. 4. C. 57-64

19. Baghbani M. IDEF0 Modeling Standard: a tool for process map drawing under requirements of ISO 9001: 2015: a case study // Journal of Modern Processes in Manufacturing and Production. Autumn 2019. Vol. 8, Issue. 4. C. 57-64
20. Denys M. Clean Architecture of Android Apps with Practical Examples. // RubyGarage : Blog about behind-the-scenes look at web and mobile development. URL: https://rubygarage.org/blog/clean-android-architecture (дата звернення 10.11.2023)
21. Save data in a local database using Room. *Android Developers.* URL: https://developer.android.com/training/data-storage/room (дата звернення 12.11.2023)
22. OkHttp Interceptor for Retrofit2. With Example. *Android Clarified.* URL: https://androidclarified.wordpress.com/2018/09/02/okhttp-interceptor-retrofit2-example/ (дата звернення 11.11.2023)
23. Koin. Injection in Android. *Koin – The pragmatic Kotlin & Kotlin Multiplatform Dependency Injection framework.* URL: https://insert-koin.io/docs/reference/koin-android/get-instances (дата звернення 10.11.2023)
24. Gson User Guide. *GitHub/Google.* URL: https://github.com/google/gson/blob/main/UserGuide.md (дата звернення 15.11.2023)
25. Madona S Wambua, Modern Android 13 Development Cookbook: Over 70 recipes to solve Android development issues and create better apps with Kotlin and Jetpack Compose. Packt Publishing Ltd., 2023. 322c.
26. Sedunov A., Kotlin In-Depth: A Guide to a Multipurpose Programming Language for Server-Side, Front-End, Android, and Multiplatform Mobile. Bpb Publications, 2022. 688c.
27. Soshin A., Kotlin Design Patterns and Best Practices - Second Edition: Build scalable applications using traditional, reactive, and concurrent design patterns in Kotlin. Packt Publishing Ltd., 2022. 356c.
28. Moskala M., Kotlin Coroutines: Deep Dive. Second Edition. Amazon Digital Services LLC – KDP Print US, 2022. 452c.

29. Späth P. Learn Kotlin for Android Development. Apress. 2019. 508 с.
30. Retrofit2. *Retrofit.* URL: https://square.github.io/retrofit/ (дата звернення 11.11.2023)
31. Guide to app architecture. *Android Developers.* URL: https://developer.android.com/topic/architecture (дата звернення 10.11.2023)
32. Build a responsive UI with ConstraintLayout. *Android Developers.* URL: https://developer.android.com/develop/ui/views/layout/constraint-layout (дата звернення 23.11.2023)

ДОДАТОК А
Планування робіт

Деталізація проекту методом SMART - це процес розбиття великих цілей проекту на більш дрібні, більш конкретні та здійсненні завдання. Метод SMART використовується для того, щоб забезпечити, щоб цілі проекту були чітко визначені, вимірювані, досяжні, актуальні та обмежені в часі. Тож застосуємо цей метод до проекту.

Таблиця А.1 – Деталізація проекту за методом SMART

Specific	Розробка мобільного додатку для моніторингу енергетичних мікромереж на основі відновлювальних джерел енергії, націленого на використання користувачами мікромереж.
Measurable	Результатом проекту має бути Android-додаток, який дозволить отримувати моніторингові дані з віддаленого серверу.
Achievable	Додаток має бути реалізований мовою Kotlin, з використанням сучасних трендів в архітектурі, та з використанням локальної бази даних для збереження даних, отриманих через API-інтерфейси.
Relevant	Програмний продукт має надавати актуальні дані поточного та прогнозованого стану енергетичної мікромережі.
Time-framed	Розробка мобільного додатку має вкластися у визначені терміни виконання, вкладаючись в календарний план кожним своїм етапом розробки.

Планування змісту робіт.

Work Breakdown Structure (WBS) або Структура розбиття робіт - це інструмент управління проектами, який розкладає весь обсяг робіт проекту на окремі, добре визначені елементи. Кожен елемент є чітко визначеним завданням чи ділянкою роботи, що може бути виміряною та керованою. Структура розбиття робіт допомагає

розглядати проект на більш дрібні, управляючі частини, що полегшує планування та виконання проекту. Цей інструмент допомагає визначити обсяг проекту, виокремити його ключові етапи та розподілити завдання між командою. WBS розробки проекту зображено на рисунку А.1.

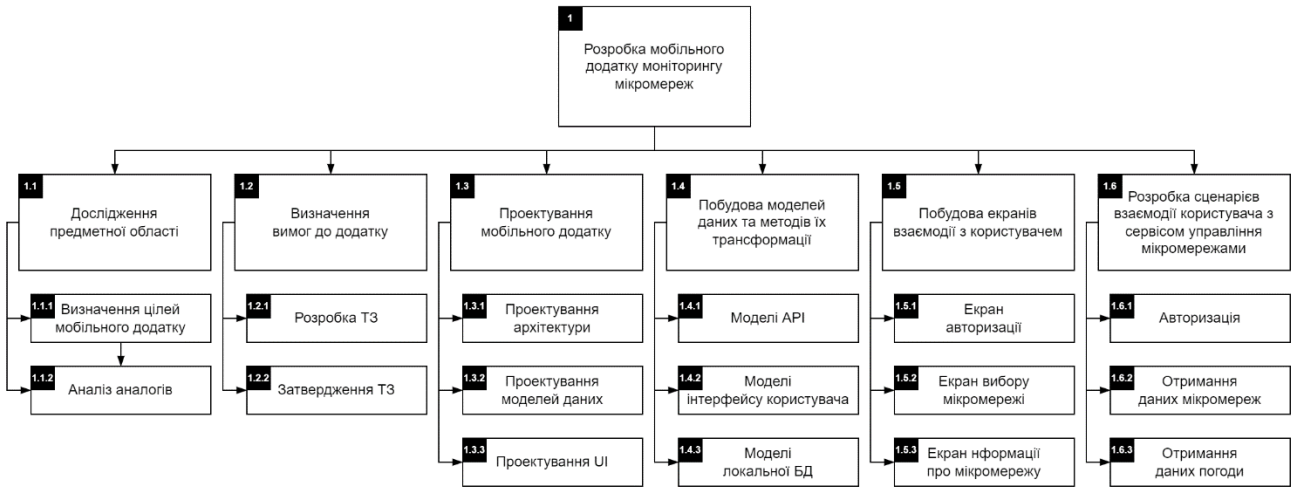

Рисунок А.1 – WBS-структура робіт по проекту

Планування структури виконавців.

Organizational Breakdown Structure (OBS) або Структура розбиття організації - це інструмент управління проектами, який відображає ієрархію та структуру організації, включаючи різні рівні управління та відділи. OBS використовується для визначення відповідальності та зв’язків між різними частинами організації та їх участю у проекті. У структурі розбиття організації можна відзначити різні рівні управління, функціональні відділи, команди чи інші групи працівників. Кожен рівень може мати свої власні відповідальності та обов'язки у рамках проекту. OBS допомагає визначити, які частини організації залучені до виконання конкретних завдань та як організаційна структура впливає на виконання проекту. OBS розробки проекту зображено на рисунку А.2. Список ролей виконавців проекту описано в таблиці А.2.

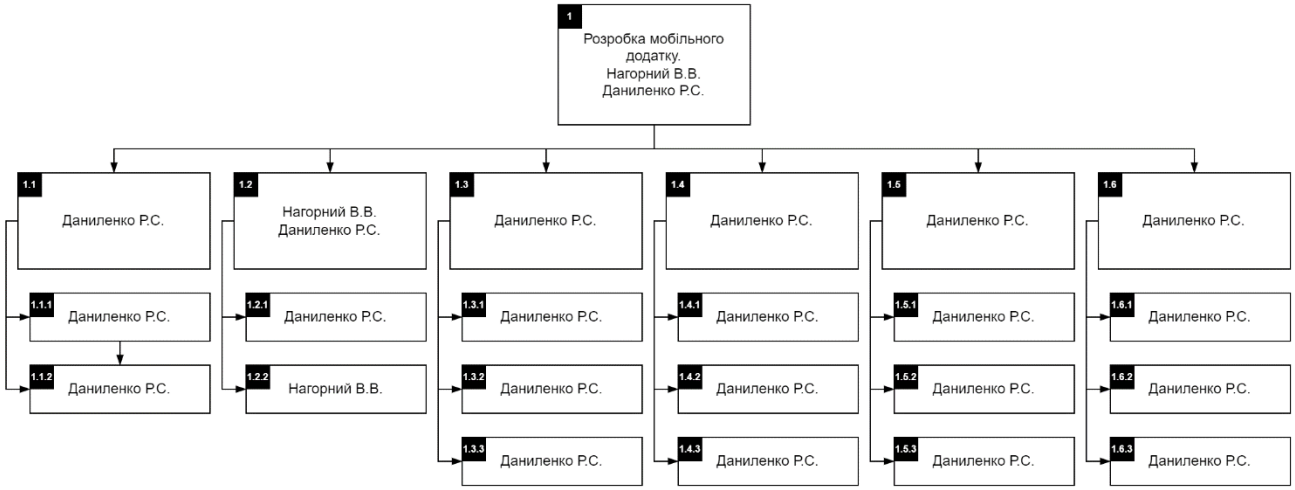

Рисунок А.2 – OBS-структура робіт по проекту

Таблиця А.2 – Ролі виконавців проекту

Роль	Ім’я	Роль на проекті
Розробник	Даниленко Р.С.	Відповідальній за проектування і розробку функціоналу проекту.
Менеджер проекту	Нагорний В.В.	Відповідальний за дотримання термінів, збору та аналізу даних.

Діаграма Ганта.

Діаграма Ганта – це інструмент проектного управління, що використовується для візуального відображення часового плану проекту. Ця діаграма подає завдання та фази проекту у вигляді стовпців на графіку за часовою шкалою. Кожен стовпець на діаграмі Ганта представляє окреме завдання чи фазу проекту, а його довжина відображає тривалість цього етапу. Горизонтальна вісь відображає час, а вертикальна вісь представляє різні завдання або фази. Кольори можуть вказувати на різні типи завдань або різні учасники проекту.

За допомогою MS Excel було побудовано діаграму Ганта, що візуалізує тривалість кожного етапу розробку мобільного додатку, визначеного на момент створення WBS. Діаграму наведено на рисунку А.3.

Назва задачі	Початок	Кінець	Тривалість
Дослідження предметної області	7.11.2023	07.11.2023	1 день
Визначення вимог до додатку	8.11.2023	9.11.2023	2 дні
Проектування мобільного додатку	10.11.2023	14.11.2023	3 дні
Побудова моделей даних та методів їх трансформації	15.11.2023	20.11.2023	4 дні
Побудова екранів взаємодії з користувачем	21.11.2023	27.11.2023	5 днів
Розробка сценаріїв взаємодії користувача з сервісом управління мікромережами	28.11.2023	4.12.2023	5 днів

Рисунок А.3 – Діаграма Ганта

Планування ризиків проекту

Ризик – це подія, яка може статися або не статися, але якщо вона станеться, то може мати позитивний або негативний вплив на проект. Якісна оцінка ризиків допомагає визначити, які з них є найбільш важливими. Під час якісної оцінки ризиків необхідно враховувати такі фактори:

– Імовірність настання ризику.

– Вплив ризику на проект, якщо він настане.

Ризики, які мають високу ймовірність настання та/або високий вплив на проект, необхідно усунути або знизити їх вплив. Кількісна оцінка ризиків допомагає визначити, наскільки ймовірний той чи інший ризик, а також оцінити його вплив на проект. Кількісна оцінка ризиків може бути виконана за допомогою різних методів, наприклад, методу експертних оцінок або методу статистичного аналізу. Кількісне та якісне оцінювання ризиків можуть бути виконані одночасно або окремо. Одночасне виконання оцінювання дає більш точне уявлення про ризики проекту. Окреме виконання оцінювання дозволяє зосередитися на конкретних аспектах ризиків. Класифікація ризиків за впливом на проект, а також ймовірність їх виникнення наведено в таблиці А.3.

Для зниження негативного впливу ризиків на проект, необхідно спланувати реагування на них. План реагування на ризики - це процес розробки заходів, які допоможуть збільшити можливості проекту і зменшити загрози для його цілей.

Таблиця А.3 – Шкала оцінювання ризиків

Оцінка	Ймовірність	Вплив	Тип
1	Дуже низька, низька	Дуже малий, малий	Прийнятні
2	Середня	Середній	Виправдані
3	Висока, дуже висока	Високий, дуже високий	Неприпустимі

Як результат планування реагування – отримується матриця ймовірності виникнення ризиків та впливу ризику. Результати вносяться в таблицю А.4. Прийнятні ризики на матриці позначаються зеленим кольором, виправдані позначаються жовтим кольором, а недопустимі – червоним.

Таблиця А.4 – Матриця впливу та ймовірностей

Ймовірність	Вплив ризику				
	Дуже малий 0.05	Малий 0.1	Середній 0.2	Великий 0.4	Дуже великий 0.8
0.9			R2(0.18)		
0.7			R7(0.14)	R1(0.28)	
0.5			R6(0.1)		
0.3		R8(0.03)		R5(0.12) R9(0.12)	
0.1			R3(0.02)	R4(0.04)	R10(0.08)

Відповідно до отриманих значень індексів ми можемо класифікувати ризики за рівнем – таблиця А.5. Ризики та стратегії реагування на них наведені у таблиці А.6.

Таблиця А.5 – Оцінювання за рівнем ризику

Назва	Межі	Ризик, що входять
Прийнятні	$0.005 \leq R \leq 0.05$	3, 4, 8
Виправдані	$0.05 < R \leq 0.14$	5, 6, 7, 9
Недопустимі	$0.14 < R \leq 0.72$	1, 2, 10

Таблиця А.6 – Ризики та стратегії реагування.

Опис	Ймовірність	Вплив	Ранг	Ступінь дії	План запобігання	План реакції
Збій апаратного забезпечення	Низька	Малий	1	Істотний	Періодично проводити перевірку апаратного забезпечення на працездатність.	Провести ремонт, або замінити у випадку, коли ремонт займає багато часу.
Збій програмного забезпечення	Низька	Великий	2	Істотний	Використання ліцензійного ПЗ, мати інстальований антивірус і регулярно його оновлювати.	Переустановлення проблемного ПЗ.
Захворювання розробника	Дуже висока	Середній	3	Ігнорувати	Члени проекту повинні мати можливість виконувати задачі один одного. Враховувати при пануванні додаткові дні для таких випадків.	Делегувати повноваження іншому розробнику, якщо терміни виконання цього вимагають.
Недотримання календарного плану	Дуже низька	Середній	1	Ігнорувати	Враховувати всі можливі затримки на етапі планування.	Переоцінити тривалості виконання етапів розробки, зробити зміни в існуючий план.

ДОДАТОК Б

Фрагменти програмного коду мобільного додатку

AndroidManifext.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission
android:name="android.permission.CHANGE_NETWORK_STATE" />

    <application
        android:name=".app.ApplicationApp"
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Micropower"
        tools:targetApi="31"
        android:usesCleartextTraffic="true">
        <activity
            android:name=".ui.MainActivity"
            android:exported="true"
            android:windowSoftInputMode="adjustResize">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

AppModule.kt

```
package com.ssu.micropower.app

import com.ssu.micropower.api.AuthInterceptor
import com.ssu.micropower.api.provideAuthApi
import com.ssu.micropower.api.provideLocationApi
import com.ssu.micropower.api.provideOkHttpClient
import com.ssu.micropower.api.provideRetrofit
```

```
import com.ssu.micropower.api.provideWeatherApi
import com.ssu.micropower.database.AppDatabase
import com.ssu.micropower.repositories.UserRepository
import com.ssu.micropower.usecases.AuthUseCase
import com.ssu.micropower.repositories.IUserRepository
import com.ssu.micropower.usecases.IAuthUseCase
import com.ssu.micropower.repositories.ILocationRepository
import com.ssu.micropower.usecases.ILocationUseCase
import com.ssu.micropower.api.INetworkManager
import com.ssu.micropower.repositories.IWeatherRepository
import com.ssu.micropower.usecases.IWeatherUseCase
import com.ssu.micropower.repositories.LocationRepository
import com.ssu.micropower.usecases.LocationUseCase
import com.ssu.micropower.api.NetworkManager
import com.ssu.micropower.repositories.WeatherRepository
import com.ssu.micropower.usecases.WeatherUseCase
import com.ssu.micropower.ui.auth.AuthViewModel
import com.ssu.micropower.ui.location.LocationViewModel
import com.ssu.micropower.ui.main.MainViewModel
import org.koin.android.ext.koin.androidApplication
import org.koin.android.ext.koin.androidContext
import org.koin.androidx.viewmodel.dsl.viewModel
import org.koin.dsl.module

val networkModule = module {
    factory { AuthInterceptor(get()) }

    factory { provideOkHttpClient(get()) }
    factory { provideAuthApi(get()) }
    factory { provideLocationApi(get()) }
    factory { provideWeatherApi(get()) }

    single { provideRetrofit(get()) }

    single<INetworkManager> { NetworkManager(androidContext()) }

    single<IUserRepository> { UserRepository(get(), get()) }
    single<ILocationRepository> { LocationRepository(get(), get()) }
    single<IWeatherRepository> { WeatherRepository(get(), get()) }

    single {
AppDatabase.getInstance(androidApplication()).localWeatherDao() }
    single {
AppDatabase.getInstance(androidApplication()).localUserDao() }
    single {
AppDatabase.getInstance(androidApplication()).localLocationDao() }
}

val useCaseModule = module {
    single<IAuthUseCase> { AuthUseCase(get(), get(), get(), get(),
get()) }
    single<ILocationUseCase> { LocationUseCase(get(), get(), get()) }
```

```
    single<IWeatherUseCase> { WeatherUseCase(get(), get()) }
}

val appModule = module {
    single { SessionManager(androidContext()) }
    viewModel { AuthViewModel(get()) }
    viewModel { MainViewModel(get(), get(), get()) }
    viewModel { LocationViewModel(get()) }
}
```

ServiceFactory.kt

```
package com.ssu.micropower.api

import com.ssu.micropower.app.Constants
import okhttp3.OkHttpClient
import okhttp3.logging.HttpLoggingInterceptor
import retrofit2.Retrofit
import retrofit2.converter.gson.GsonConverterFactory

fun provideRetrofit(okHttpClient: OkHttpClient): Retrofit =
    Retrofit.Builder()
        .baseUrl(Constants.BASE_URL)
        .client(okHttpClient)
        .addConverterFactory(GsonConverterFactory.create())
        .build()

fun provideOkHttpClient(
    authInterceptor: AuthInterceptor
) =
    OkHttpClient()
        .newBuilder()
        .addInterceptor(authInterceptor)
        .addInterceptor(
            HttpLoggingInterceptor()
                .apply {
                    level = HttpLoggingInterceptor.Level.BODY
                }
        )
        .build()

fun provideWeatherApi(retrofit: Retrofit): IWeatherService =
    retrofit.create(IWeatherService::class.java)

fun provideAuthApi(retrofit: Retrofit): IAuthService =
    retrofit.create(IAuthService::class.java)

fun provideLocationApi(retrofit: Retrofit): ILocationService =
    retrofit.create(ILocationService::class.java)
```

AuthInterceptor.kt

```
package com.ssu.micropower.api

import com.ssu.micropower.app.SessionManager
import okhttp3.Interceptor
import okhttp3.Response
import javax.inject.Inject

class AuthInterceptor @Inject constructor(
    private val sessionManager: SessionManager
) : Interceptor {
    override fun intercept(chain: Interceptor.Chain): Response {
        val token = sessionManager.getAccessToken()
        if (token != null) {
            val headers = chain.request().headers.newBuilder()
                .add("Authorization", token)
                .build()

            val request =
chain.request().newBuilder().headers(headers).build()
            return chain.proceed(request)
        }

        return chain.proceed(chain.request())
    }
}
```

ILocationService.kt

```
package com.ssu.micropower.api

import com.ssu.micropower.models.api.BaseResponse
import com.ssu.micropower.models.api.LocationDto
import com.ssu.micropower.models.api.LocationStructureDto
import retrofit2.Response
import retrofit2.http.GET
import retrofit2.http.Path

interface ILocationService {

    @GET("customers/{customerId}/locations")
    suspend fun getLocations(
        @Path("customerId") customerId: Long
    ): Response<BaseResponse<List<LocationDto>>>

    @GET("customers/{customerId}/microgrids/{locationId}/structure")
    suspend fun getLocationStructure(
        @Path("customerId") customerId: Long,
        @Path("locationId") locationId: Long
    ) : Response<BaseResponse<LocationStructureDto>>
```

```
    @GET("customers/{customerId}/microgrids/{locationId}/components")
    suspend fun getLocationComponents(
        @Path("customerId") customerId: Long,
        @Path("locationId") locationId: Long
    ) : Response<BaseResponse<List<Any>>>
}
```

ILocationUseCase.kt

```
package com.ssu.micropower.usecases

import com.ssu.micropower.api.Result
import com.ssu.micropower.models.domain.Location
import com.ssu.micropower.models.domain.LocationStructure
import kotlinx.coroutines.flow.Flow

interface ILocationUseCase {
    suspend fun getLocations(): Flow<Result<List<Location>>>

    suspend fun getLocationsLocal(): Flow<Result<List<Location>>>

    suspend fun getStructure(locationId: Long):
Flow<Result<LocationStructure>>

    suspend fun getComponents(locationId: Long):
Flow<Result<List<Any>>>
}
```

LocationUseCase.kt

```
package com.ssu.micropower.usecases

import com.google.gson.Gson
import com.google.gson.internal.LinkedTreeMap
import com.ssu.micropower.api.Result
import com.ssu.micropower.api.onFailure
import com.ssu.micropower.api.onSuccess
import com.ssu.micropower.models.api.GenerationBatteryDto
import com.ssu.micropower.models.api.GenerationSolarDto
import com.ssu.micropower.models.api.GenerationStateDto
import com.ssu.micropower.models.api.GenerationWindDto
import com.ssu.micropower.models.dao.ILocationDao
import com.ssu.micropower.models.dao.IUserDao
import com.ssu.micropower.models.domain.Location
import com.ssu.micropower.models.domain.LocationStructure
import com.ssu.micropower.models.toDao
import com.ssu.micropower.models.toDomain
import com.ssu.micropower.repositories.ILocationRepository
import kotlinx.coroutines.flow.Flow
import kotlinx.coroutines.flow.flow
import javax.inject.Inject
```

```
class LocationUseCase @Inject constructor(
    private val locationRepository: ILocationRepository,
    private val locationDao: ILocationDao,
    private val userDao: IUserDao,
) : ILocationUseCase {
    override suspend fun getLocations(): Flow<Result<List<Location>>>
{
        return flow {
            val customer = userDao.getUser()

            locationRepository
                .getLocations(customer.id)
                .onSuccess { locations ->
                    locations
                        .map { it.toDomain() }
                        .also { emit(Result.Success(it)) }
                        .map { it.toDao() }
                        .also { locationDao.insert(it) }
                }
                .onFailure {
                    emit(Result.Failure(it))
                }
        }
    }

    override suspend fun getLocationsLocal():
Flow<Result<List<Location>>> {
        return flow {
            emit(
                Result.Success(
                    locationDao
                        .getLocations()
                        .map { it.toDomain() }
                )
            )
        }
    }

    override suspend fun getStructure(locationId: Long):
Flow<Result<LocationStructure>> {
        val customer = userDao.getUser()
        return flow {
            locationRepository
                .getStructure(customer.id, locationId)
                .onSuccess { structure ->
                    structure
                        .toDomain()
                        .also {
                            emit(Result.Success(it))
                            // TODO: save to DB
                        }
```

```
                }
                .onFailure {
                    emit(Result.Failure(it))
                }
        }
    }

    override suspend fun getComponents(locationId: Long):
Flow<Result<List<Any>>> {
        val customer = userDao.getUser()
        return flow {
            locationRepository
                .getComponents(customer.id, locationId)
                .onSuccess { components ->
                    println(components)
                    // TODO:
                    val result = components
                        .filterIsInstance<LinkedTreeMap<*, *>>()
                        .map { it as LinkedTreeMap<*, *> }
                        .map {
                            Gson().fromJson(
                                Gson().toJsonTree(it),
                                when {

it.containsKey("actual_solar_generation_data") ->
GenerationSolarDto::class.java

it.containsKey("actual_wind_generation_data") ->
GenerationWindDto::class.java

it.containsKey("current_capacity_data") ->
GenerationBatteryDto::class.java

it.containsKey("current_system_state_data") ->
GenerationStateDto::class.java
                                    else -> Any::class.java
                                }
                            )
                        }
                    emit(Result.Success(result))
                }
                .onFailure {
                    emit(Result.Failure(it))
                }
        }
    }
}
```

ILocationRepository.kt

```
package com.ssu.micropower.repositories
```

```
import com.ssu.micropower.api.Result
import com.ssu.micropower.models.api.LocationDto
import com.ssu.micropower.models.api.LocationStructureDto

interface ILocationRepository {

    suspend fun getLocations(customerId: Long):
Result<List<LocationDto>>

    suspend fun getStructure(customerId: Long, locationId: Long):
Result<LocationStructureDto>

    suspend fun getComponents(customerId: Long, locationId: Long):
Result<List<Any>>
}
```

LocationRepository.kt

```
package com.ssu.micropower.repositories

import com.ssu.micropower.api.INetworkManager
import com.ssu.micropower.api.ILocationService
import com.ssu.micropower.api.Result
import com.ssu.micropower.api.consumeRequest
import com.ssu.micropower.models.api.LocationDto
import com.ssu.micropower.models.api.LocationStructureDto
import javax.inject.Inject

class LocationRepository @Inject constructor(
    private val locationService: ILocationService,
    private val networkManager: INetworkManager,
) : ILocationRepository {
    override suspend fun getLocations(customerId: Long):
Result<List<LocationDto>> {
        return networkManager.consumeRequest {
locationService.getLocations(customerId) }
    }

    override suspend fun getStructure(
        customerId: Long,
        locationId: Long
    ): Result<LocationStructureDto> {
        return networkManager.consumeRequest {
locationService.getLocationStructure(customerId, locationId) }
    }

    override suspend fun getComponents(
        customerId: Long,
        locationId: Long
    ): Result<List<Any>> {
        return networkManager.consumeRequest {
locationService.getLocationComponents(customerId, locationId) }
```

```
    }
}
```

MainViewModel.kt

```
package com.ssu.micropower.ui.main

import androidx.lifecycle.LiveData
import androidx.lifecycle.MutableLiveData
import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.ssu.micropower.api.Result
import com.ssu.micropower.api.getMessages
import com.ssu.micropower.api.onFailure
import com.ssu.micropower.api.onSuccess
import com.ssu.micropower.models.api.GenerationBatteryDto
import com.ssu.micropower.models.api.GenerationSolarDto
import com.ssu.micropower.models.api.GenerationStateDto
import com.ssu.micropower.models.api.GenerationWindDto
import com.ssu.micropower.models.domain.GenerationItem
import com.ssu.micropower.models.domain.Location
import com.ssu.micropower.models.domain.LocationInfo
import com.ssu.micropower.models.domain.LocationStructure
import com.ssu.micropower.models.domain.Weather
import com.ssu.micropower.models.toDate
import com.ssu.micropower.models.toGenerationItem
import com.ssu.micropower.models.toGenerationItemDevice
import com.ssu.micropower.usecases.IAuthUseCase
import com.ssu.micropower.usecases.ILocationUseCase
import com.ssu.micropower.usecases.IWeatherUseCase
import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.async
import kotlinx.coroutines.awaitAll
import kotlinx.coroutines.ensureActive
import kotlinx.coroutines.flow.collect
import kotlinx.coroutines.flow.combine
import kotlinx.coroutines.launch
import javax.inject.Inject

class MainViewModel @Inject constructor(
    private val authUseCase: IAuthUseCase,
    private val locationUseCase: ILocationUseCase,
    private val weatherUseCase: IWeatherUseCase
) : ViewModel() {

    private val _message = MutableLiveData<String>()
    val message: LiveData<String> = _message

    private val _locations = MutableLiveData<List<Location>>()
    val locations: LiveData<List<Location>> = _locations

    private val _locationInfo = MutableLiveData<LocationInfo>()
```

```
    val locationInfo: LiveData<LocationInfo> = _locationInfo

    private val _weather = MutableLiveData<Weather>()
    val weather: LiveData<Weather> = _weather

    private val _logout = MutableLiveData<Boolean>()
    val logout: LiveData<Boolean> = _logout

    fun getLocations() {
        viewModelScope.launch(Dispatchers.IO) {
            ensureActive()

            locationUseCase
                .getLocations()
                .collect { result ->
                    result
                        .onSuccess { _locations.postValue(it) }
                        .onFailure {
_message.postValue(it.getMessages()) }
                }
        }
    }

    fun getLocationStructure(location: Location) {
        viewModelScope.launch(Dispatchers.IO) {
            ensureActive()

            val (resA, resB) = awaitAll(
                async { locationUseCase.getStructure(location.id) },
                async { locationUseCase.getComponents(location.id) }
            )
            resA.combine(resB) { structureResult, componentsResult ->
                when {
                    structureResult is Result.Success &&
componentsResult is Result.Success -> {
                        val structure = structureResult.value as
LocationStructure

                        val items = (componentsResult.value as
List<*>)
                            .filter { it is GenerationSolarDto || it
is GenerationBatteryDto || it is GenerationWindDto }
                            .map { component ->
                                when (component) {
                                    is GenerationSolarDto ->
component.toGenerationItem(
                                        structure.solarPanels.map {
it.toGenerationItemDevice() }
                                    )

                                    is GenerationWindDto ->
component.toGenerationItem(
```

```
                                        structure.windMills.map {
it.toGenerationItemDevice() }
                                    )

                                    is GenerationBatteryDto ->
component.toGenerationItem(
                                        structure.accumulators.map {
it.toGenerationItemDevice() }
                                    )

                                    else -> {}
                                }
                            } as List<GenerationItem>

                        val status = componentsResult.value
                            .find { it is GenerationStateDto } as
GenerationStateDto

                        _locationInfo.postValue(
                            LocationInfo(
                                location = location,
                                time =
status.systemState.time.toDate(),
                                consumption =
status.systemState.consumption,
                                sell = status.systemState.sell,
                                purchase =
status.systemState.purchase,
                                generation = items
                            )
                        )
                    }

                    else -> {
                        structureResult.onFailure {
_message.postValue(it.getMessages()) }
                        componentsResult.onFailure {
_message.postValue(it.getMessages()) }
                    }
                }
            }.collect()
        }
    }

    fun getWeather(location: Location) {
        viewModelScope.launch(Dispatchers.IO) {
            ensureActive()

            weatherUseCase
                .getWeather(location.id, location.coordinates)
                .collect { result ->
                    result
```

```
                        .onSuccess { _weather.postValue(it) }
                        .onFailure {
_message.postValue(it.getMessages()) }
                }

        }
    }

    fun logout() {
        viewModelScope.launch(Dispatchers.IO) {
            ensureActive()

            authUseCase
                .logout()
                .collect { result ->
                    result
                        .onSuccess { _logout.postValue(it) }
                        .onFailure {
_message.postValue(it.getMessages()) }
                }

        }
    }
}
```

MainFragment.kt

```
package com.ssu.micropower.ui.main

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.Fragment
import androidx.fragment.app.setFragmentResultListener
import androidx.navigation.findNavController
import androidx.recyclerview.widget.LinearSnapHelper
import com.google.android.material.snackbar.Snackbar
import com.ssu.micropower.R
import com.ssu.micropower.databinding.FragmentMainBinding
import com.ssu.micropower.models.domain.Location
import com.ssu.micropower.ui.adapters.GenerationAdapter
import com.ssu.micropower.ui.adapters.WeatherAdapter
import com.ssu.micropower.ui.location.LocationFragment
import org.koin.androidx.viewmodel.ext.android.viewModel
import java.text.SimpleDateFormat
import kotlin.math.roundToInt

class MainFragment : Fragment() {

    private lateinit var _binding: FragmentMainBinding
```

```
    private val viewModel: MainViewModel by viewModel()

    private lateinit var adapter: GenerationAdapter

    private lateinit var adapterWeather: WeatherAdapter

    private var isInit = false

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        _binding = FragmentMainBinding.inflate(inflater, container,
false)
        return _binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState:
Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        adapter = GenerationAdapter()
        adapterWeather = WeatherAdapter()

        _binding.apply {
            lifecycleOwner = this@MainFragment

            lstComponents.adapter = adapter
            LinearSnapHelper().attachToRecyclerView(lstComponents)

            lstWeather.adapter = adapterWeather
            LinearSnapHelper().attachToRecyclerView(lstWeather)

            viewModel.message.observe(viewLifecycleOwner) {
                isLoading = false
                isLoadingWeather = false
                Snackbar.make(container, it,
Snackbar.LENGTH_LONG).show()
            }

            viewModel.locations.observe(viewLifecycleOwner) {
locations ->
                println(locations)

                locations
                    .takeIf { it.isNotEmpty() }
                    ?.first()
                    ?.let {
                        viewModel.getLocationStructure(it)
                        viewModel.getWeather(it)
                        isInit = true
```

```
                }
            }

            viewModel.locationInfo.observe(viewLifecycleOwner) {
locationInfo ->
                btnLocations.text = locationInfo.location.name
                txtTime.text =
SimpleDateFormat(getString(R.string.date_time_normal)).format(location
Info.time)
                txtConsumption.text = getString(R.string.value_kwh,
locationInfo.consumption)
                txtSell.text = getString(R.string.value_kwh,
locationInfo.sell)
                txtPurchase.text = getString(R.string.value_kwh,
locationInfo.purchase)

                adapter.setItems(locationInfo.generation)
                lstComponents.smoothScrollToPosition(0)

                isLoading = false
            }

            viewModel.weather.observe(viewLifecycleOwner) {
                it.current.apply {
                    txtWeatherTemp.text =
getString(R.string.weather_temp_value, temperature)
                    txtWeatherPrecipitation.text =
getString(R.string.weather_precipitation_value, precipitation)
                    txtWeatherPressure.text =
getString(R.string.weather_pressure_value, pressure)
                    txtWeatherWindSpeed.text =
getString(R.string.weather_wind_speed_value, windSpeed)

                    imgWindDirection.rotation =
(windDirection.toFloat() / 45f).roundToInt() * 45f + 90f
                }

                adapterWeather.setItems(it.hourly)

                isLoadingWeather = false
            }

            viewModel.logout.observe(viewLifecycleOwner) {
                activity

?.findNavController(R.id.nav_host_fragment_container)
                    ?.navigateUp()
            }

            btnLocations.setOnClickListener {
```

```
setFragmentResultListener(LocationFragment.SELECT_LOCATION) { _,
bundle ->
                    bundle
                        .takeIf {
it.containsKey(LocationFragment.LOCATION_ITEM) }

?.getSerializable(LocationFragment.LOCATION_ITEM)
                        ?.let { it as Location }
                        ?.let {
                            isLoading = true
                            isLoadingWeather = true
                            viewModel.getLocationStructure(it)
                            viewModel.getWeather(it)
                        }
                }

                activity

?.findNavController(R.id.nav_host_fragment_container)
                    ?.navigate(
                        R.id.action_mainFragment_to_locationFragment
                    )
            }

            btnLogout.setOnClickListener {
                viewModel.logout()
            }
        }

        if (!isInit) {
            _binding.isLoading = true
            _binding.isLoadingWeather = true
            viewModel.getLocations()
        }
    }
}
```

MainActivity.kt

```
package com.ssu.micropower.ui

import android.content.Context
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.inputmethod.InputMethodManager
import com.ssu.micropower.R

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
```

```
    }

    fun hideKeyboard() {
        val imm = getSystemService(Context.INPUT_METHOD_SERVICE) as
InputMethodManager
        imm.hideSoftInputFromWindow(currentFocus?.windowToken, 0)
    }
}
```

fragment_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    tools:ignore="ContentDescription">

    <data>

        <import type="android.view.View" />

        <variable
            name="isLoading"
            type="Boolean" />

        <variable
            name="isLoadingWeather"
            type="Boolean" />

    </data>

    <androidx.constraintlayout.widget.ConstraintLayout
        android:id="@+id/container"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <androidx.constraintlayout.widget.Guideline
            android:id="@+id/guideWeather"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:orientation="horizontal"
            app:layout_constraintGuide_end="196dp" />

        <androidx.recyclerview.widget.RecyclerView
            android:id="@+id/lstComponents"
            android:layout_width="0dp"
            android:layout_height="0dp"
            android:layout_marginBottom="@dimen/margin_x2"
            android:background="@drawable/bg_main"
            android:orientation="horizontal"

app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"
```

```
        app:layout_constraintBottom_toTopOf="@id/lstWeather"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:listitem="@layout/list_item_generation_solar"/>

    <ImageButton
        android:id="@+id/btnLogout"
        style="@style/Button.Image"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="@dimen/margin_x2"
        android:background="@null"
        android:padding="@dimen/margin_x2"
        android:src="@drawable/ic_logout"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:tint="@color/color_primary" />

    <TextView
        android:id="@+id/txtTime"
        style="@style/Label.Date"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/txtConsumption"
        style="@style/Label.Status"
        android:text="@string/value_default_kwh"
        app:drawableStartCompat="@drawable/ic_consumption"
        app:layout_constraintEnd_toStartOf="@id/txtSell"
        app:layout_constraintHorizontal_chainStyle="packed"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@id/txtTime" />

    <TextView
        android:id="@+id/txtSell"
        style="@style/Label.Status"
        android:text="@string/value_default_kwh"
        app:drawableStartCompat="@drawable/ic_sell"
        app:layout_constraintEnd_toStartOf="@id/txtPurchase"
        app:layout_constraintStart_toEndOf="@id/txtConsumption"
        app:layout_constraintTop_toBottomOf="@id/txtTime" />

    <TextView
        android:id="@+id/txtPurchase"
        style="@style/Label.Status"
        android:text="@string/value_default_kwh"
        app:drawableStartCompat="@drawable/ic_buy"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toEndOf="@id/txtSell"
```

```
            app:layout_constraintTop_toBottomOf="@id/txtTime" />

        <TextView
            android:id="@+id/txtWeatherTemp"
            style="@style/Label.Weather.Main"
            android:text="@string/weather_temp_value_default"
            app:layout_constraintBottom_toTopOf="@id/guideWeather"

app:layout_constraintEnd_toStartOf="@id/txtWeatherPrecipitation"
            app:layout_constraintHorizontal_chainStyle="spread"
            app:layout_constraintStart_toStartOf="parent" />

        <TextView
            android:id="@+id/txtWeatherPrecipitation"
            style="@style/Label.Weather.Main"
            android:text="@string/weather_precipitation_value_default"
            app:layout_constraintBottom_toTopOf="@id/guideWeather"

app:layout_constraintEnd_toStartOf="@id/txtWeatherPressure"
            app:layout_constraintStart_toEndOf="@id/txtWeatherTemp" />

        <TextView
            android:id="@+id/txtWeatherPressure"
            style="@style/Label.Weather.Main"
            android:text="@string/weather_pressure_value_default"
            app:layout_constraintBottom_toTopOf="@id/guideWeather"

app:layout_constraintEnd_toStartOf="@id/txtWeatherWindSpeed"

app:layout_constraintStart_toEndOf="@id/txtWeatherPrecipitation" />

        <TextView
            android:id="@+id/txtWeatherWindSpeed"
            style="@style/Label.Weather.Main"
            android:text="@string/weather_wind_speed_value_default"
            app:layout_constraintBottom_toTopOf="@id/guideWeather"
            app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintStart_toEndOf="@id/txtWeatherPressure" />

        <TextView
            android:id="@+id/lblWeatherTemp"
            style="@style/Label.Hint"
            android:text="@string/weather_temp_label"
            app:layout_constraintBottom_toTopOf="@id/txtWeatherTemp"
            app:layout_constraintEnd_toEndOf="@id/txtWeatherTemp"
            app:layout_constraintStart_toStartOf="@id/txtWeatherTemp"
/>

        <TextView
            android:id="@+id/lblWeatherPrecipitation"
            style="@style/Label.Hint"
```

```
            android:text="@string/weather_precipitation_label"

app:layout_constraintBottom_toTopOf="@id/txtWeatherPrecipitation"

app:layout_constraintEnd_toEndOf="@id/txtWeatherPrecipitation"

app:layout_constraintStart_toStartOf="@id/txtWeatherPrecipitation" />

        <TextView
            android:id="@+id/lblWeatherPressure"
            style="@style/Label.Hint"
            android:text="@string/weather_pressure_label"

app:layout_constraintBottom_toTopOf="@id/txtWeatherPressure"
            app:layout_constraintEnd_toEndOf="@id/txtWeatherPressure"

app:layout_constraintStart_toStartOf="@id/txtWeatherPressure" />

        <TextView
            android:id="@+id/lblWeatherWind"
            style="@style/Label.Hint"
            android:text="@string/weather_wind_speed_label"

app:layout_constraintBottom_toTopOf="@id/txtWeatherWindSpeed"
            app:layout_constraintEnd_toEndOf="@id/txtWeatherWindSpeed"

app:layout_constraintStart_toStartOf="@id/txtWeatherWindSpeed" />

        <ImageView
            android:id="@+id/imgWindDirection"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:rotation="90"
            android:src="@drawable/ic_wind_direction"

app:layout_constraintBottom_toBottomOf="@id/txtWeatherWindSpeed"

app:layout_constraintStart_toEndOf="@id/txtWeatherWindSpeed"
            app:layout_constraintTop_toTopOf="@id/txtWeatherWindSpeed"
/>

        <Button
            android:id="@+id/btnLocations"
            style="@style/Button.Location"
            android:maxWidth="240dp"
            android:text="@string/button_location"
            app:layout_constraintBottom_toBottomOf="@id/guideWeather"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="@id/guideWeather" />
```

```
<com.google.android.material.progressindicator.CircularProgressIndicat
or
            android:id="@+id/prbLoading"
            style="@style/ProgressBar.Loading"
            android:visibility="@{isLoading ? View.VISIBLE :
View.GONE, default=visible}"
            app:layout_constraintBottom_toBottomOf="@id/btnLocations"
            app:layout_constraintStart_toEndOf="@id/btnLocations"
            app:layout_constraintTop_toTopOf="@id/btnLocations" />

        <androidx.recyclerview.widget.RecyclerView
            android:id="@+id/lstWeather"
            android:layout_width="0dp"
            android:layout_height="0dp"
            android:layout_marginHorizontal="@dimen/margin_x2"
            android:layout_marginTop="@dimen/margin_x12"
            android:layout_marginBottom="@dimen/margin_x2"
            android:orientation="horizontal"
            android:overScrollMode="never"
            android:visibility="@{isLoadingWeather ? View.GONE :
View.VISIBLE, default=visible}"

app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="@id/guideWeather"
            tools:listitem="@layout/list_item_weather"/>


<com.google.android.material.progressindicator.CircularProgressIndicat
or
            android:id="@+id/prbLoadingWeather"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:indeterminate="true"
            android:visibility="@{isLoadingWeather ? View.VISIBLE :
View.GONE, default=visible}"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="@id/lstWeather" />

    </androidx.constraintlayout.widget.ConstraintLayout>

</layout>
```

list_item_generation_solar.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    tools:ignore="ContentDescription">

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:padding="16dp">


        <androidx.constraintlayout.widget.Guideline
            android:id="@+id/guideTop"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:orientation="horizontal"
            app:layout_constraintGuide_begin="80dp" />

        <androidx.constraintlayout.widget.Guideline
            android:id="@+id/guideCenter"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:orientation="vertical"
            app:layout_constraintGuide_percent="0.50" />

        <TextView
            android:id="@+id/txtPowerMaxHint"
            style="@style/Label.Hint"
            android:text="@string/generator_power_max"
            app:layout_constraintEnd_toEndOf="@id/prbValueCurrent"
            app:layout_constraintStart_toEndOf="@id/prbValueCurrent"
            app:layout_constraintTop_toTopOf="@id/guideTop" />

        <TextView
            android:id="@+id/txtPowerMax"
            style="@style/Label.Value"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/value_default_kwh"
            app:layout_constraintEnd_toEndOf="@id/prbValueCurrent"
            app:layout_constraintStart_toEndOf="@id/prbValueCurrent"
            app:layout_constraintTop_toBottomOf="@id/txtPowerMaxHint"
/>

        <ImageView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@drawable/ic_generator_solar"
            app:layout_constraintBottom_toBottomOf="@id/txtPowerMax"
            app:layout_constraintEnd_toStartOf="@id/prbValueCurrent"
            app:layout_constraintTop_toTopOf="@id/txtPowerMax"
            app:tint="@color/color_text_hint" />
```

```
<com.google.android.material.progressindicator.CircularProgressIndicat
or
            android:id="@+id/prbValueCurrent"
            style="@style/ProgressBar.Generator"
            android:layout_marginTop="@dimen/margin_x8"
            android:progress="66"
            app:indicatorColor="@color/color_primary"
            app:indicatorSize="240dp"
            app:layout_constraintEnd_toEndOf="@id/guideCenter"
            app:layout_constraintStart_toStartOf="@id/guideCenter"
            app:layout_constraintTop_toBottomOf="@id/guideTop" />


<com.google.android.material.progressindicator.CircularProgressIndicat
or
            android:id="@+id/prbValueForecast"
            style="@style/ProgressBar.Generator"
            android:progress="33"
            app:indicatorColor="@color/color_secondary"
            app:indicatorSize="204dp"

app:layout_constraintBottom_toBottomOf="@id/prbValueCurrent"
            app:layout_constraintEnd_toEndOf="@id/guideCenter"
            app:layout_constraintStart_toStartOf="@id/guideCenter"
            app:layout_constraintTop_toTopOf="@id/prbValueCurrent" />

        <ImageView
            android:id="@+id/imgConditionCurrent"
            style="@style/ImageCondition"
            android:src="@drawable/ic_weather_0"
            app:layout_constraintBottom_toTopOf="@id/txtValueCurrent"
            app:layout_constraintEnd_toEndOf="@id/guideCenter"
            app:layout_constraintStart_toStartOf="@id/guideCenter"
            app:layout_constraintTop_toTopOf="@id/prbValueCurrent"
            app:layout_constraintVertical_chainStyle="packed"
            app:tint="@color/color_primary" />

        <TextView
            android:id="@+id/txtValueCurrent"
            style="@style/Label.Value.Normal"
            android:text="@string/value_default_kwh"
            app:layout_constraintBottom_toTopOf="@id/txtDateCurrent"
            app:layout_constraintEnd_toEndOf="@id/guideCenter"
            app:layout_constraintStart_toStartOf="@id/guideCenter"

app:layout_constraintTop_toBottomOf="@id/imgConditionCurrent" />

        <TextView
            android:id="@+id/txtDateCurrent"
            style="@style/Label.Date.Small"
            app:layout_constraintBottom_toTopOf="@id/pnlDivider"
```

```
            app:layout_constraintEnd_toEndOf="@id/guideCenter"
            app:layout_constraintStart_toStartOf="@id/guideCenter"
            app:layout_constraintTop_toBottomOf="@id/txtValueCurrent"
/>

        <View
            android:id="@+id/pnlDivider"
            android:layout_width="0dp"
            android:layout_height="1dp"
            android:layout_marginHorizontal="@dimen/margin_x12"
            android:layout_marginVertical="@dimen/margin_x1"
            android:background="@color/color_text_hint"
            app:layout_constraintBottom_toTopOf="@id/txtDateForecast"
            app:layout_constraintEnd_toEndOf="@id/prbValueForecast"

app:layout_constraintStart_toStartOf="@id/prbValueForecast"
            app:layout_constraintTop_toBottomOf="@id/txtDateCurrent"
/>


        <TextView
            android:id="@+id/txtDateForecast"
            style="@style/Label.Date.Small"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            app:layout_constraintBottom_toTopOf="@id/txtValueForecast"
            app:layout_constraintEnd_toEndOf="@id/guideCenter"
            app:layout_constraintStart_toStartOf="@id/guideCenter"
            app:layout_constraintTop_toBottomOf="@id/pnlDivider" />

        <TextView
            android:id="@+id/txtValueForecast"
            style="@style/Label.Value.Normal"
            android:text="@string/value_default_kwh"

app:layout_constraintBottom_toTopOf="@id/imgConditionForecast"
            app:layout_constraintEnd_toEndOf="@id/guideCenter"
            app:layout_constraintStart_toStartOf="@id/guideCenter"
            app:layout_constraintTop_toBottomOf="@id/txtDateForecast"
/>

        <ImageView
            android:id="@+id/imgConditionForecast"
            style="@style/ImageCondition"
            android:src="@drawable/ic_weather_0"

app:layout_constraintBottom_toBottomOf="@id/prbValueCurrent"
            app:layout_constraintEnd_toEndOf="@id/guideCenter"
            app:layout_constraintStart_toStartOf="@id/guideCenter"
            app:layout_constraintTop_toBottomOf="@id/txtValueForecast"
            app:tint="@color/color_secondary" />
```

```
        <com.google.android.material.chip.ChipGroup
            android:id="@+id/pnlDevices"
            style="@style/ListChip"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@id/prbValueCurrent">

            <include layout="@layout/chip_device" />

        </com.google.android.material.chip.ChipGroup>

    </androidx.constraintlayout.widget.ConstraintLayout>

</layout>
```