МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Секція інформаційно-комунікаційних технологій

ВИПУСКНА РОБОТА

на тему:

«Кросплатформний мобільний додаток-довідник для ОС Android та iOS»

Завідувач
випускаючої кафедри **Довбиш А.С.**

Керівник роботи **Шутилєва О.В.**

Студента гр.Ін-71 **Порошин Д.А.**

СУМИ 2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютерних наук

Затверджую_____

Зав. кафедрою Довбиш А.С.

"_____" _____2021 р.

ЗАВДАННЯ
до випускної роботи

Студента четветого курсу, групи ІН-71 спеціальності «122 – Комп'ютерних наук» денної форми навчання Порошина Дмитра Андрійовича.

Тема: " Кросплатформний мобільний додаток-довідник для ОС Android та iOS"

Затверджена наказом по СумДУ

№_____ від _____2021 р.

Зміст пояснювальної записки: 1) літературний огляд за обраною тематикою роботи; 2) методика виконання поставлених задач; 3) практична реалізація мобільного додатку;

Дата видачі завдання "_____"_____2021 р.

Керівник випускної роботи _____ Шутилєва О.В.

Завдання прийняв до виконання _____ Порошин Д.А.

РЕФЕРАТ

ЗАПИСКА: 124 сторінки, 32 рисунка, 10 джерел літератури, 1 додаток.

ОБ'ЄКТ ДСЛІДЖЕННЯ – технологія створення крос-платформного мобільного додатку для отримання необхідної інформації.

МЕТА РОБОТИ – проектування та розробка довідника агрохімічних препаратів через реалізацію мобільного додатку за допомогою технології Flutter.

МЕТОДИ ДОСЛІДЖЕННЯ − база даних Hive, технологія Flutter, клієнт-серверна технологія, програмна платформа WordPress.

РЕЗУЛЬТАТ − було обрано оптимальні технології для розробки мобільного додатку; архітектура та користувацький інтерфейс забезпечують необхідний функціонал, тестування додатку показало його готовність до запуску.

FLUTTER, DART, ANDROID, IOS, МОБІЛЬНИЙ ДОДАТОК
ФРЕЙМОРК, ДОДАТОК-ДОВІДНИК

ЗМІСТ

ВСТУП

Мобільні пристрої це один з найпопулярніших видів електронних девайсів. Щорічно виробляється близько одного мільярда таких пристроїв. Такий великий об’єм зумовлений тим що без такого девайсу неможливо увити сучасне життя. Мобільні пристрої використовують для спілкування з іншими людьми та обміну медіа даними, перегляду відео або створення нотаток.

Однією з провідних функцій мобільних пристроїв стала пошук інформації. Майже в будь якому куточку світу, користувач має можливість вийти в мережу та знайти необхідну інформацію. Девайс не займає багато місця у кишені, тому він завжди з нами.

З кожним роком пристрої розвиваються, стають більш потужними. Зараз вони здатні на паралельні обчислювання великих потоків даних та зберігати гігабайти інформації для миттєвого доступу. Тому мобільні прилади часто можуть замінити собою громіздкі персональні комп’ютери.

Основним завдання даної роботи є створення мобільного додатку, що дозволить користувачам шукати інформацію про агрохімікати. Основною функцією цього додатку є робота без доступу до інтернету та кросплатформність для максимального охоплення ринку.

Актуальність роботи полягає у відсутності єдиного додатку для отримання інформації про всі препарати які виготовляються або імпортуються в країну. На фоні стрімкого розвитку сільськогосподарського сектору в Україні та важливості цієї сфери постає необхідність у покращенні робочих умов аграрії та задача у створенні єдиної бази даних всіх препаратів.

1. ЛІТЕРАТУРНИЙ ОГЛЯД ЗА ОБРАНОЮ ТЕМАТИКОЮ РОБОТИ

По-перше, необхідно розглянути операційні системи Android та iOS. Важливо розуміти принцип роботи ОС, для правильного вибору інструменту розробки та уникнення помилок при роботі з ними. Проаналізуємо особливості архітектури даних систем.

По-друге, розглянемо мови програмування, що є нативними для Android та iOS. Окремим пунктом необхідно виділити кросплатформену розробку. Навести переваги та недоліки такого методу в порівнянні з традиційним підходом.

1.1 Операційна система Android

Android – це мобільна операційна система, заснована на модифікованій версії ядра Linux та іншого програмного забезпечення з відкритим кодом, призначена в основному для сенсорних мобільних пристроїв, таких як смартфони та планшети [1, с. 34]. В останні роки, операційна система Android також підтримує електронні книги, мультимедійні програвачі, «розумні» годинники, нетбуки та інших пристрої. Розробкою операційної системи займається консорціум розробників, відомий як Open Handset Alliance за фінансової підтримки компанії Google.

Android є найпоширенішою операційною системою (ОС) для мобільних пристроїв. Дана система має безліч особливостей, які роблять її популярною і привабливою для великої кількості користувачів по всьому світу.

Операційна система Android є невимогливою та здатна працювати на різних конфігураціях. Починаючі від флагманських пристроїв з багатоядерними та потужними процесорами і закінчуючи бюджетними рішеннями. Через це більшість світових виробників встановлюють на свої

пристрої саме цю ОС, оскільки інші програмні продукти призначені для певної апаратної складової, що відповідає певній специфікації.

Своїй гнучкості Android завдячує, тому, що система побудована на ядрі Linux. Це ядро має відкритий програмний код, це дозволяє розробникам адаптувати його під свою апаратну платформу. Найновіша версія Android може бути запущений на пристроях, що мають об'єм оперативної пам'яті менше 2 Гб. Система не вимагає наявності високопродуктивного процесора і може працювати на пристроях з 4 процесорними ядрами та частотами від 1500 МГц. З 2017 року, вийшла платформа Android Go, що є більш простою версією, а ніж стандартний дистрибутив Android. Вона призначається для бюджетних та супер бюджетних пристроїв, які мають менше ніж 2 Гб RAM, але підтримує всі основні функції.

Основною мовою програмування для Android є Java – це об'єктно-орієнтована мова програмування розроблена у 1995 році та розробляється компанією Oracle. Це мова програмування створена за принципом WORA(Write once, run anywhere), що означає, що скомпільований код Java може працювати на всіх платформах, що підтримують Java, без необхідності перекомпіляції [2, с 27-29].

Для неї було створено багато різних бібліотек та API, як самою компанією Google, так і сторонніми розробниками. Це забезпечує легкість у написанні програм та дає змогу використовувати всі можливості ОС Android. Починаючі від 2019 року, пріоритетною мовою програмування є Kotlin, тому Java поступово втрачає свою популярність на ринку мобільних пристроїв. Kotlin у свою чергу має декілька переваг:

- код більш лаконічний, тобто ви зможете писати і підтримувати меншу кількість коду;
- код більш безпечний, оскільки він є Null safety;
- інтеграція з Android Studio.

Android Studio – це інтегроване середовище розробки, що створено Google для розробки програмного забезпечення під операційну систему Android [3].

Ці фактори та швидкий ріст аудиторії мови Kotlin позитивно впливають на популярність не лише серед малих компаній та стартап проектів, а й великих корпорацій.

Android структурований у формі стеку програмного забезпечення, що включає програми, операційну систему, середовище виконання, проміжне програмне забезпечення, сервіси та бібліотеки. Архітектуру операційної системи представлено візуально на рисунку 1. Кожен елемент стеку та відповідні компоненти в кожному рівні добре інтегровані та ретельно налаштовані, щоб забезпечити оптимальне середовище розробки та виконання додатків для мобільних пристроїв.

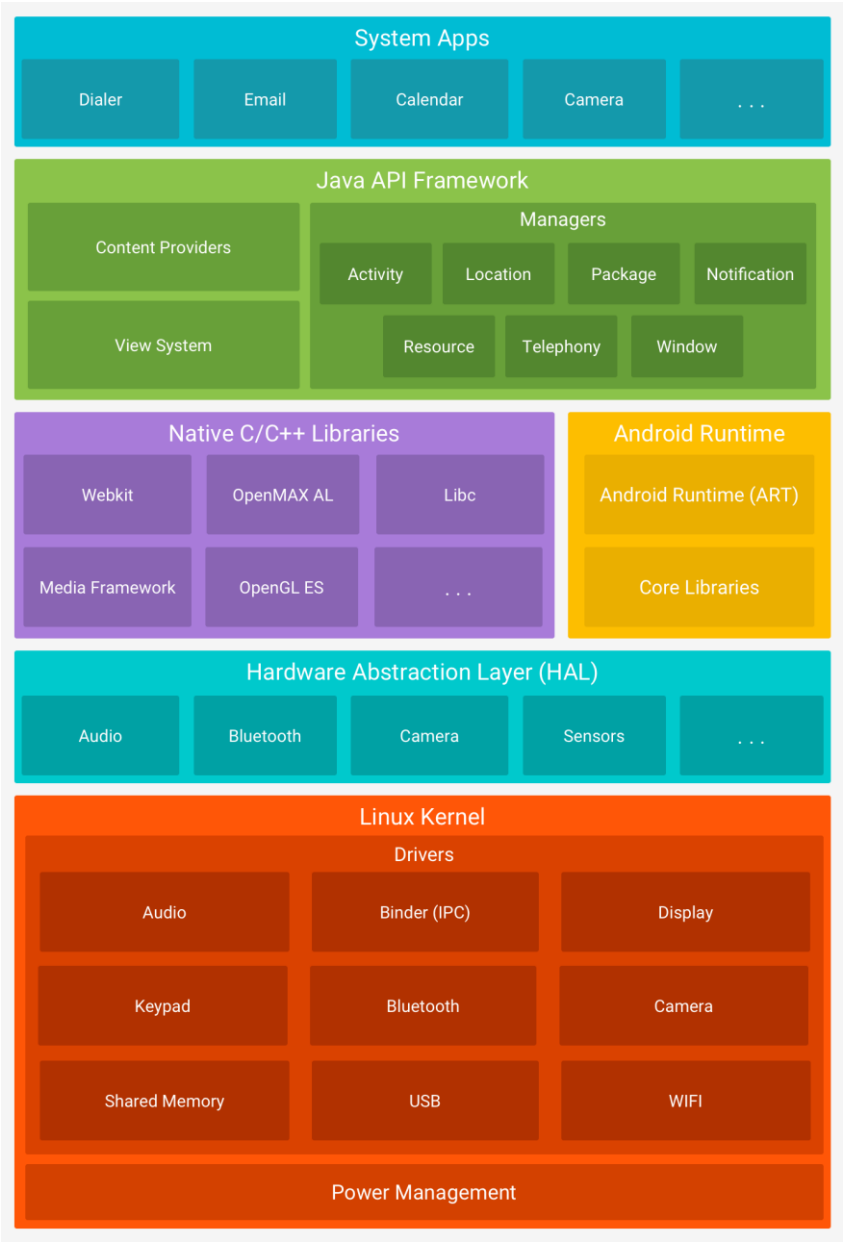

Рисунок 1 – Архітектура ОС Android

Внизу стеку програмного забезпечення Android розташоване ядро Linux. Воно забезпечує рівень абстракції між апаратним забезпеченням пристрою та верхніми шарами операційної системи.

Коли додаток Android будується, він компілюється у проміжний формат байт-коду, цей формат іменований як DEX. Коли програма завантажується на пристрій користувача, Android Runtime (ART) використовує процес, який називається компіляцією Ahead-of-Time (AOT). Цей процес необхідний для

конвертації байт-код до інструкцій, що будуть зрозумілі для центрального процесора.

Рівень апаратної абстракції (HAL) надає стандартні інтерфейси, які надають апаратні можливості пристрою вищому рівню Java API. HAL складається з безлічі бібліотечних модулів, кожен з яких реалізує інтерфейс для певного типу апаратних компонентів, таких як камера або модуль Bluetooth. Коли API середовища здійснює виклик для доступу до апаратного забезпечення пристрою, система Android завантажує бібліотечний модуль для цього апаратного компонента.

Багато основних системних компонентів та служб Android, таких як ART або HAL, побудовані з коду, що вимагає бібліотек, написаних на C та C ++. Платформа Android надає API, для надання функціоналу деяких з цих власних бібліотек додаткам.

Java API Framework − це набір служб та сервісів, що разом формують середовище, в якому працюють додатки Android. Це середовище забезпечує коректну роботу програми

У верхній частині стеку архітектури Android знаходяться додатки. Це можуть бути як вбудовані програми, що постачаються разом з операційною системою (наприклад, календар та годинник), так і сторонні програми, встановлені користувачем.

1.2 Операційна система iOS

IOS – це мобільна операційна система, створена та розроблена компанією Apple лише для свої продуктів [4, с 1-2]. Ця операційна система встановлена на мобільні пристрої компанії, такі як iPhone та iPod Touch. Раніше, термін iOS включав також версії, що працюють на iPad, але в 2019 році була введена назва iPadOS, так як версія операційної системи для планшетних комп’ютерів мала значні відмінності. Потрібно зазначити, що це друга за

популярністю мобільна операційна система у світі після Android. iOS виступила основою для трьох інших операційних систем компанії Apple, а саме iPadOS, tvOS і watchOS. Операційна система є закритої та її вихідний код належить компанії розробнику, хоча, треба зазначити, деякі його частини є відкритими для розробників.

Операційна система iOS є пропрієтарною, це означає що вона може працювати лише на певних пристроях, для яких вона була розроблена. ОС здатна працювати лише з певними процесорами та компонентами, що передбачені компанією Apple, а закритий вихідний код не дає можливості стороннім розробникам додавати підтримку інших апаратних платформ.

У такого закритого підходу є свої переваги. По-перше, оскільки операційна система розрахована лише для декількох апаратних платформ, то її швидкодія та стабільність зростає. Це пояснюється тим, що розробник має змогу оптимізувати код системи, а кількість пристрої які необхідно підтримувати значно менша, а ніж кількість під управлінням Android. По-друге, так як кількість пристроїв обмежена та контролюється, то можна забезпечити тривалий термін підтримки. Це означає, що користувач буде мати останню версію iOS протягом більше ніж п’яти років, якщо опиратися на компанію Apple. В свою чергу, аналогічний показник на Android становить лише декілька років.

Основною мовою програмування під операційну систему iOS є Swift – це універсальна багатокомпонентна мова програмування, розроблена Apple та спільнотою у 2014 році [5]. Swift у першу чергу була розроблена як заміна попередньої мови програмування Objective-C, а також має зворотну сумісність з неї.

Нова мова зберегла взаємодії з Objective-C, що дозволяє використовувати всі бібліотеки, розроблені для продуктів Apple, та дає можливість розробникам відносно швидко та безболісно перенести існуючи програми на нову мову програмування. Swift також отримав чудову інтеграцію

з іншими інструментами розробки від Apple, а саме інтеграцію з Xcode та фреймворком Cocoa Touch.

Xcode – це інтегроване середовище розробки компанії Apple для операційної системи macOS [6, с. 3]. Широко використовується для розробки програмного забезпечення для macOS, iOS, iPadOS, watchOS або tvOS.

Можна виділити основні переваги мови Swift:

- не великий, але водночас лаконічний та зрозумілий код;
- висока швидкодія та безпека у порівнянні з застарілим Objective-C;
- менша кількість необхідної пам’яті та її автоматичний менеджмент.

Усе це, разом зі швидким розвитком робить Swift чудовою заміною, тому більшість розробників використовують її, як основну мову програмування для пристроїв Apple.

Архітектура iOS складається з кількох різних програмних шарів, кожен з яких забезпечує коректну роботу програмного забезпечення, що працює на вершині операційної системи. Ці шари зручно зобразити у вигляді рисунку(див. рис. 2).

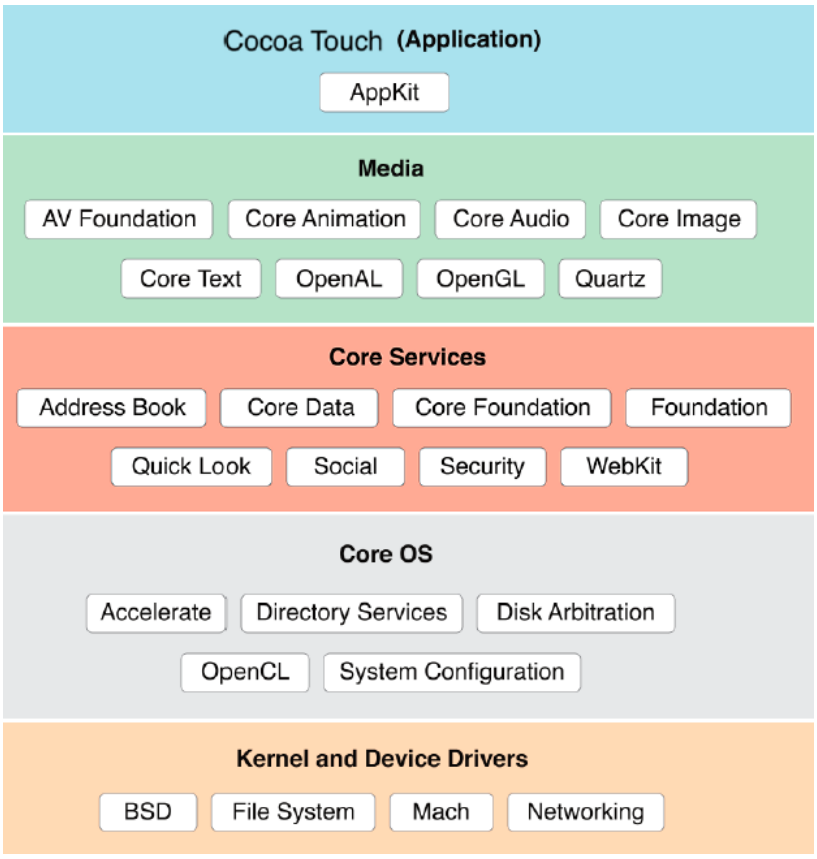

Рисунок 2 – Архітектура ОС iOS

Core OS – це нижній шар стеку iOS, який знаходиться безпосередньо у верхній частині апаратного забезпечення пристрою. Рівень забезпечує різноманітні сервісів, включаючи мережевий зв'язок низького рівня, доступ до зовнішніх аксесуарів та основні функції операційної системи, такі як управління пам'яттю, робота файлової системи та потоки.

Вище розташований Core service, який забезпечує доступ до функцій Core OS, що використовуються у верхніх шарах. Він містить такі фреймворки, як Address Book Framework, WebKit, Security та інші.

Media Service надає програмному забезпеченню аудіо, відео, інструменти для роботи з анімацією та графічними можливостями. Media Service, також як і Core service містить низку фреймворків, які можна використовувати під час розробки програми.

Cocoa Touch знаходиться на вершині стеку. Він відповідає за показ користувацького інтерфейсу програми та реагування на дії користувача. Окрім

того, більшість функцій, що визначають взаємодію з користувача з операційною, таких як Notification Center, full-screen mode та Auto Save, реалізовані на цьому шарі.

1.3 Принцип роботи крос-платформних додатків

Розробка крос-платформних додатків − це створення одного додатка з єдиною кодовою базою, який може працювати на різних операційних системах на від міну від нативної розробки, де необхідно розробляти різні версії додатків для кожної платформи.

Рушійною силою для розробки крос-платформних додатків є створення програмного забезпечення, яке добре працює на більш ніж одній операційній системи. Основна мета – охопити якомога ширшу аудиторію.

Під час розробки крос-платформних програм, можна швидко запустити програмне забезпечення на різних платформах. Вихідний код пишеться один раз для всіх платформ. Це означає, що не потрібно наймати окрему команду розробників програмного забезпечення для кожної платформи, оскільки запуск та оновлення програмного забезпечення за допомогою різноманітних крос-платформних інструментів розробки.

Можна виділити основні переваги крос-платформних підходу при розробці програмного продукту: це дешево, швидко, та легко підтримувати у майбутньому.

Оскільки розробкою крос-платформного додатку займається одна команда одразу на декілька платформ, то необхідність винаймати людей під кожну окрему операційну систему відпадає. Завдяки цьому, можна зменшити ресурси на розробку у декілька разів, в залежності від кількості ОС, що необхідні замовнику.

Швидкість розробки крос-платформного додатку зазвичай помітно вища, а ніж традиційний підхід. Це обумовлено ризиками та менеджментом.

Кількість ризиків, які можуть виникнути при розробці зменшується, що зумовлено кількістю команд при розробці, чим їх менше, тим менше число потенційних проблем. Менеджмент при цьому стає простішим, менше часу витрачається на дзвінки, обговорення та інше.

Єдина кодова база означає те, що такий продукт легко та дешево підтримувати. Достатньо додати нову функцію в код і вона з'явиться на всіх платформах. Необхідність у реалізації цієї функції на кожній окремій платформі зникає. Це позитивно також позитивно впливає і на термін підтримки.

Однак крос-платформний підхід має і свої недоліки, які необхідно знати, коли обираєте спосіб розробки. Основними є: повільніша швидкість роботи, обмежена функціональність, більші вимоги до розробників.

Швидкість роботи у більшості випадків повільніша, ніж у нативних додатків. Причина може залежати від інструменту розробки, але у свої більшості це зумовлено додатковим рівнем абстракції між операційною системою та кодом програми. Цей рівень виступає як інтерпретатор, що переводить код додатку у зрозумілий для системи та надає доступ до її складових. Треба зазначити, що різниця у швидкості помітна лише в складних задачах, що передбачає складну роботу з великими об'ємами даних.

Оскільки крос-платформні інструменти розробки не мають повного доступу до компонентів або апаратної частини операційної системи, як приклад робота з камерою, або мікрофоном, то може виникнути проблема у роботі з ними. У таких випадках, стає необхідність у написанні нативного коду, якщо така можливість передбачена інструментом розробки, або використання сторонніх бібліотек.

Розробка під декілька платформ, вимагає знати особливості кожної з них, це необхідно для досягнення найкращої швидкої та правильної роботи програмного забезпечення. Нерідко, бувають випадки, коли один код працює по різному на деяких платформах. Збільшення кількості операційних систем

збільшує кількість пристроїв у яких можуть бути різні розміри екранів, тому потрібно приділяти більшу увагу користувацьким інтерфейсам для коректного відображення інформації на дисплеї. Всі ці фактори вимагають від працівника додаткових навичок та знань, а роботодавець, нерідко, вимагає досвіду роботи з традиційними інструментами створення мобільних додатків, а саме знання мови Java або Kotlin для розробки під Android та мови Swift для написання коду для платформи iOS.

1.4 Постановка задачі

Однією з провідних галузей в Україні є сільське господарство. Воно забезпечує населення країни якісними та безпечними продуктами харчування, що дозволяє державі економити на імпорті закордонної продукції.

За типами сільське господарство поділяється на рослинництво та тваринництво. В України переважає саме рослинництво, завдяки великий площі орних земель та родючому ґрунту. За статистикою, їх площа складає більше ніж 70 % від усіх сільськогосподарських угідь.

Найпопулярніші зернові культури це – пшениця, жито, ячмінь, кукурудза та гречка. Україні посідає одне з провідних місць по виробництву цих зернових культур, поступаючись лише деяким країнам Європи. Це робить країну важливим гравцем на ринку експорту зернових культур, а зернове господарство найважливішою в аграрному секторі.

На сьогоднішній час, продаж зерна є основним джерелом іноземної валюти для України. Основними імпортерами українського зерна є Єгипет, Іспанія, Нідерланди та Китай. За інформацію 2019 року країна отримала близько 7,2 мільярдів доларів від продажу зерна і з кожним роком цей показник поступово збільшується. Тому дуже важливо розвивати цю галузь та збільшувати обсяги врожаю.

Для збільшення врожайності, більшість аграріїв використовують різні хімічні добрива. Вони захищають рослини від різних типів хвороб та шкідників, які можуть знищити левову частку врожаю та серйозно вдарити по фінансовому становищу невеликих аграріїв.

Використання добрив та оприскувачів потребує деяких знань про культуру, адже кожний препарат призначений для певної сільськогосподарської культури. Потрібно знати, які сполуки входять до складу, від яких шкідників захищає, спосіб приготування та інші аспекти препарату.

З пошуком інформації про той чи інший препарат інколи можуть виникнути складнощі – це досить часто вимагає часу. Офіційні сайти деяких виробників не відповідають сучасним вимога. Відсутність пошуку потрібного препарату, застарілий інтерфейс, непристосованість під мобільні пристрої. Останній пункт – серйозний недолік, так як аграрії часто працюють в полі, де відсутній доступ до комп’ютера.

Посіви, у своїй більшості, знаходяться у віддалених районах нашої країни. З цього випливає, що зв'язок, з навколишнім світом у таких регіонах гірший, а ніж у великих містах – це ускладнює пошук інформації безпосередньо на робочому місці, у полі або в дорозі. Тому використання інтернет ресурсів у такому випадку є незручним, а інколи неможливим.

У агрофірм, також може постати проблема з постачанням препарату, особливо, якщо це стосується оптових закупівель, адже інтернет та фізичні магазини рідко володіють великими запасами товару. Тому виникає необхідність придбання продукту безпосередньо у постачальника або виробника – це вимагає додаткового часу.

Отже, виникає необхідність у базі даних, яка буде містити інформацію про препарати, їх опис, застосування, види та застереження. Створення бази контактів виробників і постачальників товару.

Постає задача створення електронного довідника, за допомогою якого, робітники зможуть легко шукати інформацію про препарат у польових умовах, не використовуючи для цього веб-сайти. Реалізація цього довідника має бути у вигляді додатку на мобільні пристрої, адже смартфон завжди є під рукою.

Основної перевагою мобільного додатку над інтернет сайтом є можливість створити локальне сховище інформації. Це означає, що додатком, можна користуватися навіть коли відсутнє підключення до інтернету, а коли інтернет з'єднання стабільне, автоматично оновлювати сховище до актуальної версії.

Також, є сенс додати інформації не тільки, про добрива та препарати, а й про різні види культур та їх сорти. Щоб користувач зміг отримати всю необхідну для ного інформації не виходячи з додатку.

Компанії постачальники, що імпортують закордонні препарати, зможуть розміщувати їх у базі даних поряд з вітчизняним продуктом – це дає користувачу можливість порівняти декілька виробів та обрати той, що найбільше задовольняє його потреби.

У свою чергу, для компаній буде можливість розмістити свої контактні дані у додатку, щоб клієнт зміг зв'язатися з компанією виробником або постачальником та придбати бажаний товар без пошуку в інтернеті.

2. МЕТОДИКА ВИКОНАННЯ ПОСТАВЛЕНИХ ЗАДАЧ

Перед початком розробки необхідно правильно сформулювати список вимог до майбутньої програми. Завдяки йому можна створити конкуренто спроможний продукт, який не буде поступатися функціоналом. Тому важливо зробити аналіз аналогічного програмного забезпечення. Виділити слабкі сторони, та зробити висновки, як їх можна уникнути та знайти переваги і реалізувати їх у додатку.

Вибір інструменту вирішення задачі залежить від поставленого списку вимог. В залежності від цього списку можна обрати різні методи які будуть максимально оптимальними в даному випадку. На основі сформованого аналізу конкурентів, необхідно зробити вибір програмної платформи, що буде використовуватися при розробці.

2.1 Аналіз конкурентів

На даний момент додаток має декількох конкурентів на ринку. Серед них є як закордонні аналоги так і вітчизняні. Найпопулярнішими на українському ринку є «Сингента Украина», «Corteva Agriscience UA» та «Summit-Agro Ukraine».

Потрібно обрати основні критерії під час оцінки додатків. Першим критерієм буде офлайн доступ до бази даних. Це важливій критерій, так як нерідко доступ до інформації необхідний у регіонах, де відсутнє інтернет покриття, або не має стабільного покриття, що робіть використання додатку неможливим. Наступний критерій зручний та сучасний інтерфейс програми, адже інтуїтивне керування програмою допомагає користувачу швидко освоїтися та розібратися з функціями додатку. Останній критерій – це наявність контактів для зв’язку з виробником або дистриб’ютором, щоб отримати додаткову інформацію про препарат або зробити замовлення.

Окремо розглянемо недоліки та переваги кожного з додатків, виділимо основні функції та зробимо невеликий висновок.

Першим розглянемо додаток «Сингента Украина», розроблений однойменною компанією. Він дає доступ до інформації про препарати, створені самою компанією.

Головне вікно програми має лаконічний та актуальний дизайн (див. рис. 3). Немає зайвих елементів які б заважали при користуванні, тому зразу можна приступати до роботи.

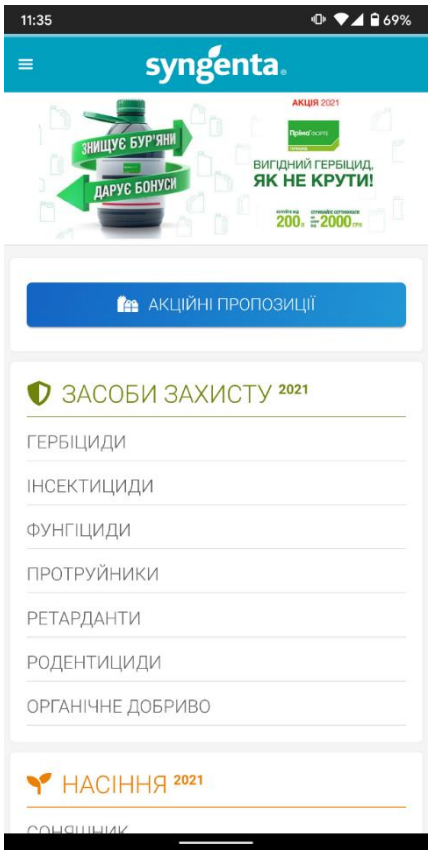

Рисунок 3 – Головне вікно програми «Сингента Украина»

Під час користування додатком, виникають незручності. Бокове меню виконано у вигляді списку з великою кількістю елементів, тому для перегляду всіх пунктів, необхідно прогортати увесь список. Також відсутній глобальний пошук, він з’являється лише тоді, коли відкриваєш конкретний розділ. Дизайн

деяких категорій відрізняється від інших, незважаючи на те, що вони мають однакову структуру, що негативно позначається на зручності користування.

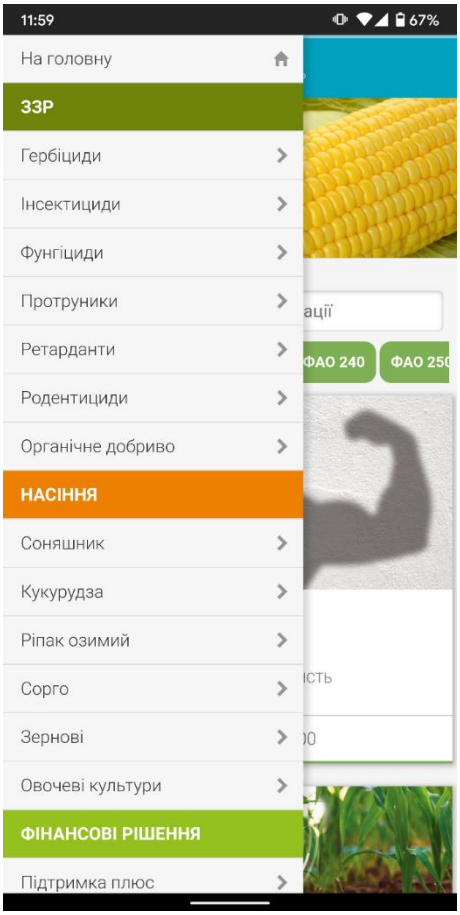

Рисунок 4 – Бокове меню програми «Сингента Украина»

Натомість, програма має всю необхідну інформацію про кожний з препаратів, для яких культур підходить та як його застосовувати. Додатково є розділ про різні види культур, який містить короткі дані про кожну та розділ-бібліотеку (див. рис. 5) з базовою інформацією про шкідників, хвороби та калькулятор, для обчислення необхідної кількості поживних речовин.

Контакти містять не лише відомості про головний офіс компанії, а й про всіх дистриб’юторів з телефонами та адресами (див. рис. 5), також тут розташований пошук, тому знайти представництво у своєму місті легко.

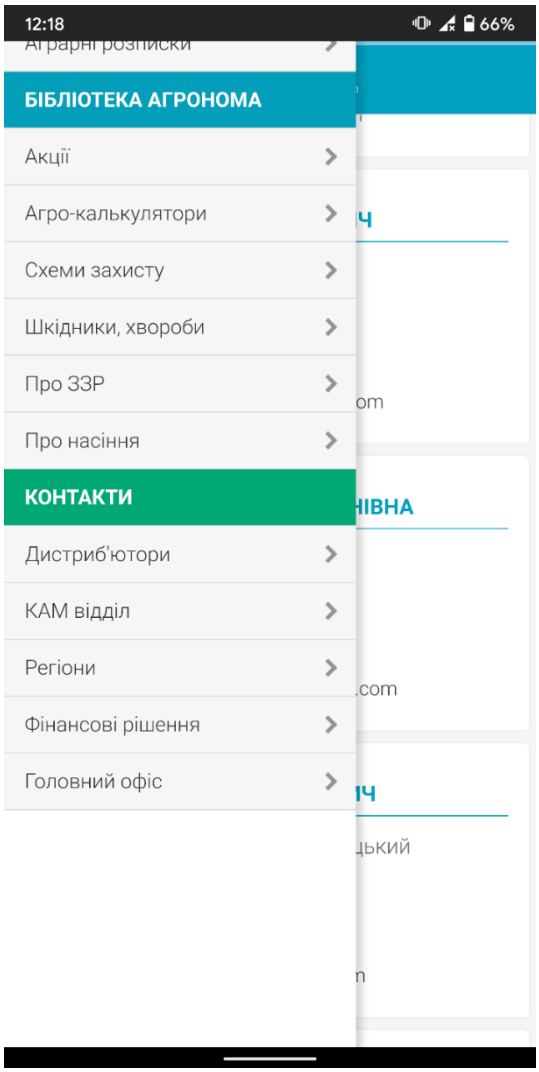

Рисунок 5 – Розділи контакти та бібліотека програми «Сингента Украина»

Основні переваги:

- вичерпна інформація про препарати;
- додаткова бібліотека;
- усі необхідні контактні дані;
- лаконічний дизайн;
- офлайн доступ до програми.

Основні недоліки:

- незручне бокове меню додатку;
- відсутність глобального пошуку по всім препаратам.

Отже, «Сингента Украина» – це програма з усіма необхідними відомостями для аграріїв, яка має лише декілька недоліків у зручності користування.

Наступним розглянемо додаток «Corteva Agriscience UA». Програма має зручне головне вікно (див. рис. 6). Всі розділи доступні одразу, не потрібно гортати сторінку, щоб побачити увесь доступний функціонал. Кожний пункт має свій знак, завдяки якому легко розпізнати розділ, не читаючи його назви.

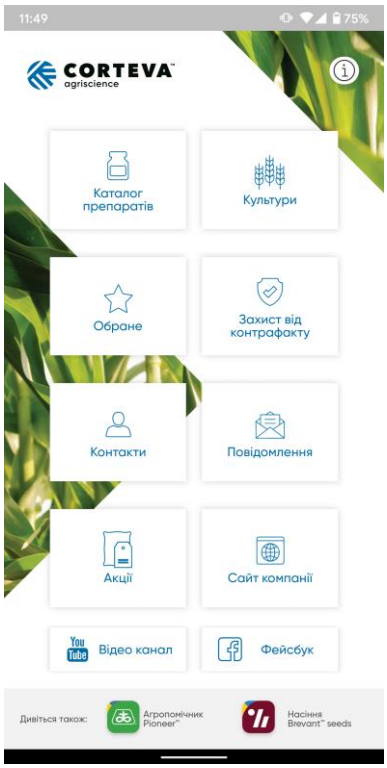

Рисунок 6 – Головне вікно програми «Corteva Agriscience UA»

Користуватися програмою легко та зручно. Все інтуїтивно зрозуміло, переходи між вікнами та розділами логічні. Зовнішній вигляд відповідає всім сучасним нормам дизайну.

Виникають проблеми з використанням фільтру та пошуку. Для застосування фільтру необхідно вводити пошукові дані власноруч (див. рис. 7) що не зручно, особливо якщо не пам’ятаєш назву культури або діючої речовини, реалізація у формі списку набагато зручніша у такій ситуації . Хоча

фільтр і має автодоповненя, але це не дає користувачу повної картини того, які дані він може використовувати для сортування.

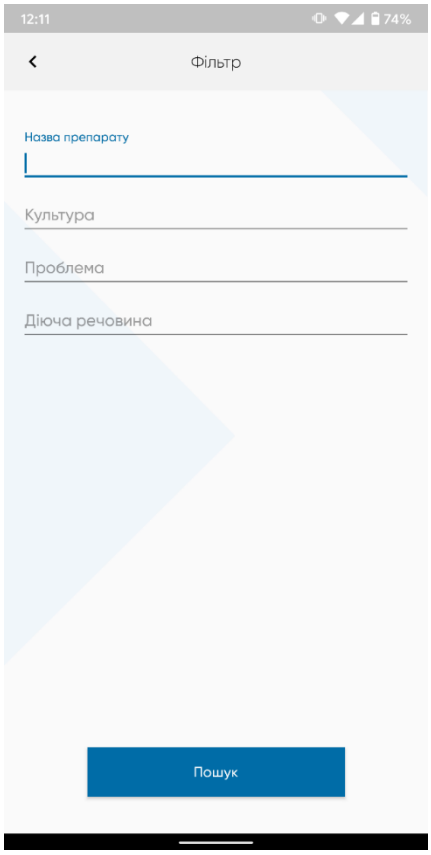

Рисунок 7 – Вікно фільтрування програми «Corteva Agriscience UA»

Програма має функцію додання обраних препаратів у окремий список, тому якщо ви часто використовуєте інформацію про один, або декілька препаратів, то ви можете користуватися саме цим списком – це зручно та практично. Оремо можна виділити розділ з контактами, він має наочну карту регіонів та представництв, а також фотокартки керівників.

Програма не має додаткових розділів з інформацією про насіння або хвороби, для цього потрібно завантажувати додаткові програми, що не зовсім комфортно, так як вимагає постійно переходити між декількома додатками.

Основні переваги:

- Всі необхідні дані про препарат
- Зручний розділ з контактами

- Список обраних препаратів
- Офлайн доступ до додатку
- Сучасний дизайн

Основні недоліки:

- Незручний фільтр та пошук
- Додаткова інформація відокремлена в окремий додаток

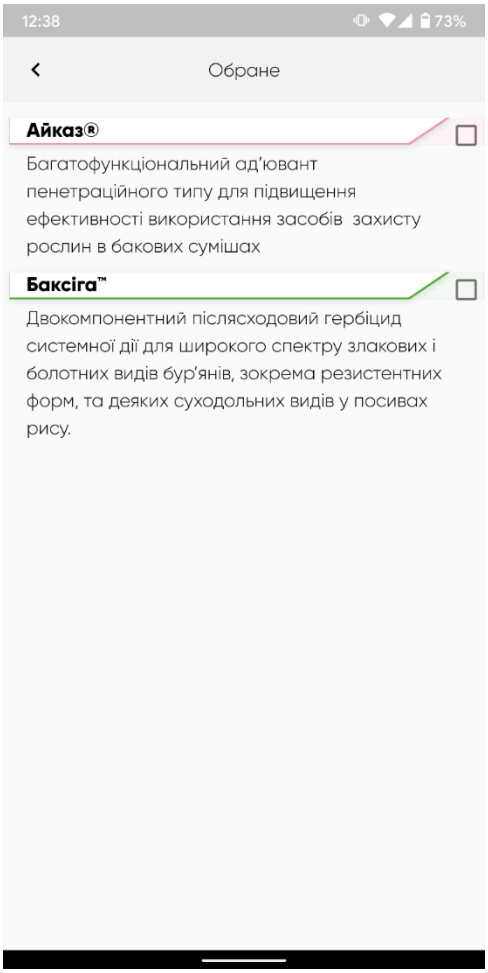

Рисунок 8 – Список обраних препаратів програми «Corteva Agriscience UA»

Отже, «Corteva Agriscience UA» непогана програма, сконцентрована лише на необхідній для користувача інформації. Про це свідчить відсутність додаткових розділив, що робіть розмір програми меншим, а ніж у конкурентів та окремий список з обраним.

Останнім додатком розглянемо «Summit-Agro Ukraine». Це найменш популярна програма в порівняні з іншими двома.

Головне вікно за своєю структурою схоже на «Corteva Agriscience UA» (див. рис. 9). Угорі розташоване поле для пошуку, що дуже зручно, можна одразу почати пошук. Програма виконує пошуковій запит по всім розділам, а не лише по обраному, а викликати його можна з будь якого вікна. Також, як і у попередньо розглянутого «Corteva Agriscience UA» є список обраних препаратів, але на відміно від конкурента, «Summit-Agro Ukraine» дозволяє додавати до цього списку ще й культури.

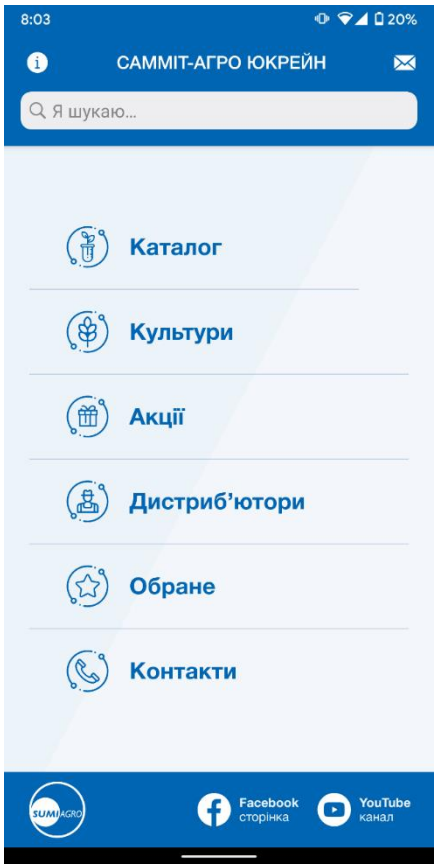

Рисунок 9 – Головне вікно програми «Summit-Agro Ukraine»

Дизайн програми викликає приємні враження. Він виглядає сучасно, а незначні анімації додають динаміки під час користуванням. Кожний розділ та категорія мають своє зображення.

Користуватися програмою легко та просто. Питання виникають лише до розділів з контактами та дистриб’юторами. Вони мають однаковий вигляд (див. рис. 10 та 11), тому легко можна переплутати в якому вікні знаходишся.

Рисунок 10 – Розділ дистриб’ютори програми «Summit-Agro Ukraine»

Рисунок 11 – Розділ контакти програми «Summit-Agro Ukraine»

Основні переваги:

- Список обраних препаратів
- Офлайн доступ до додатку
- Зручний пошук по всім розділам
- Приємний дизайн, наявність простих анімацій
- Зручне навігаційне меню

Основні недоліки:

- Відсутність фільтру
- Контакти та Дистриб’ютори дуже схожі між собою
- Відсутність додаткових розділів

Отже, додаток «Summit-Agro Ukraine» як і «Corteva Agriscience UA» сконцентрований лише на необхідній для користувача інформації. Пропонує глобальний пошук та привабливий дизайн, натомість, розділ з контактами реалізований не найкращим чином та потребує покращень.

Підводячи підсумок, всі розглянуті додатки відповідають поставленим вимогам. Вони мають гарний дизайн, яким просто користуватися, база даних має всю необхідну інформацію про препарати та доступна в офлайн режимі. Проте, більшість з них, мають не зручну фільтрацію інформації та сконцентровані лише на власних виробах, тому якщо використовувати товар від різних виробників, потрібно мати декілька встановлених програм – це головний недолік цих додатків.

На основі виконаного аналізу, формуємо список конкретних вимог для майбутнього проекту.

1. Офлайн доступ до додатку.
2. Каталог з препаратами.
3. Сучасний та зручний дизайн.
4. Вікно фільтрації препаратів.
5. Вікно пошуку.

6. Доступ до контактів агрофірм або дистриб’юторів.
7. Сторінка з новинами агросфери у країни.
8. Додатковий розділ з гібридами агрокультур.
9. Відображення базової інформації про погоду за геолокацією користувача.

2.2 Огляд інструментів вирішення задачі

Існує декілька мобільних операційних систем. Найпоширенішими з них є Android и iOS. Згідно статистиці за 2020 рік вони займають аж 99 відсотків усього ринку мобільних пристроїв. З них 74 відсотків припадають на платформу Android та 24 відсотки на iOS і лише 1 відсоток складають інші операційні системи.

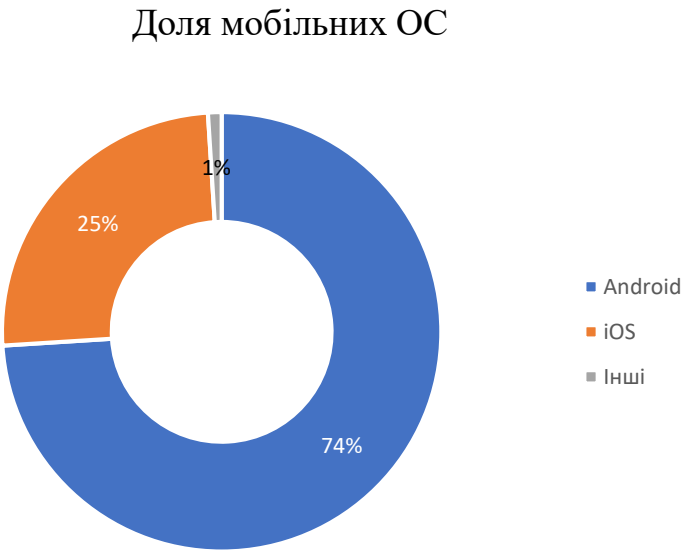

Рисунок 12 – Діаграма «Доля мобільних ОС»

Як зазначалося раніше, існує два способи розробки програмного забезпечення, а саме:

- Використання нативних інструментів
- Розробка за допомогою кросплатформних інструментів

Про використання нативних інструментів ішлося у розділах «Операційна система Android» та «Операційна система iOS» під операційні системи Android та iOS відповідно, а про різницю між нативною та кросплатформною розробкою розповідалось у розділі «Принцип роботи крос-платформних додатків», тому ми не будемо фокусувати увагу на цих пунктах, а одразу розглянемо кросплатформні інструменти та їх види.

Існує безліч інструментів для створення кросплатформного додатку. Беручі до уваги статистику популярних запитів на ресурсі «Stack Overflow» можна виділити список найбільш актуальних фреймворків. Розглянемо найбільш популярні рішення, а саме Flutter, React Native та Xamarin.

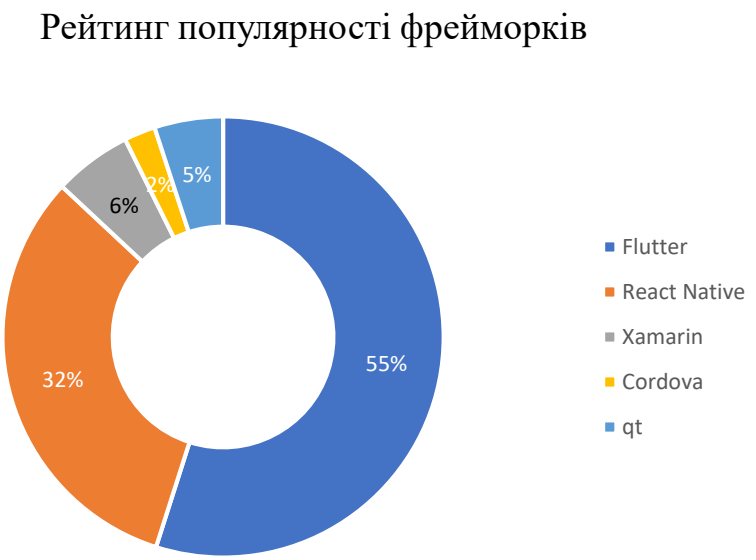

Рисунок 13 – Діаграма «Рейтинг популярності фреймворків»

2.2.1 Flutter

Flutter – це фреймворк, створений Google на мові Dart, що має відкритий кодом. Він створювався як інструмент для розробки крос-платформних додатків [7].

Dart – це мова програмування, розроблена Google, для створення серверних та настільних додатків [8, с. 3-4]. Dart – об'єктно-орієнтована мова,

заснована на класах із синтаксисом у стилі C. Може компілюватися як у власний код, так і в JavaScript.

Flutter package – це бібліотека, що створена для фреймворку Flutter, яка дозволяє використовувати готові рішення, створені іншими розробниками [9].

Flutter plugin – це плагін, створеній для фреймворку Flutter. Дає змогу взаємодіяти з системними компонентами та використовувати можливості нативного коду [9].

Особливістю фреймворку є те, що він має власний графічний рушій, тому він сам займається побудовою елементів інтерфейсу програми, контролює жести та анімацію. Це гарно відображається на його швидкодії, адже взаємодія з інтерфейсом відбувається напряму, без додаткових програмних прошарків.

Взаємодія з системними сервісами реалізована за допомогою Platform Channels. Це спеціальні канали, що слугують для передачі інформації між самим фреймворком та нативним кодом, що дає розробникам широкий спектр можливостей у використанні нативних бібліотек або функцій, які не підтримуються Flutter.

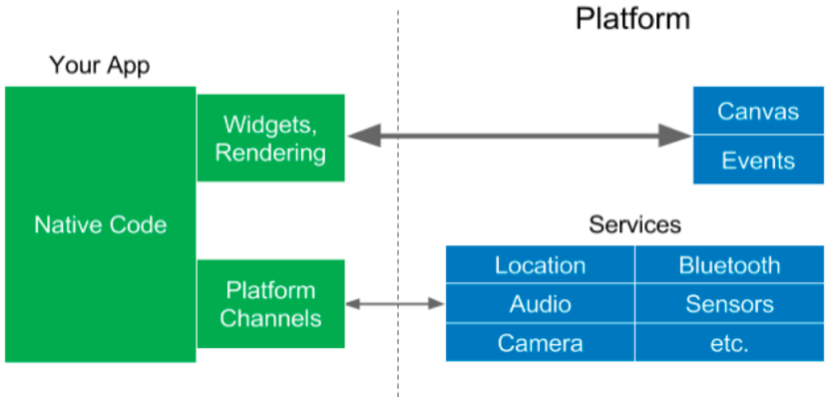

Рисунок 14 – Принцип роботи фреймворку Flutter

Незважаючи на те, що Flutter з’явився відносно недавно, він має всі функції для створення робочої програми, а швидкий та стабільний розвиток

розширює список робочих платформ не лише на Android та iOS, а й на Web рішення, що дає великі перспективи використання фреймворку.

2.2.2 React Native

React Native був створений на основі іншого фреймворку React та використовую мову Java Script, розроблений компанією Facebook та спільноти, має відкритий код.

Він не має власного графічного двигуна для виведення зображення, а використовує OEM елементи, які вбудовані в операційну систему, для побудови інтерфейсу. Щоб код програми, написаний на Java Script міг взаємодіяти з системними компонентами було створено спеціальний Bridge, він слугує адаптером між двома мовами.

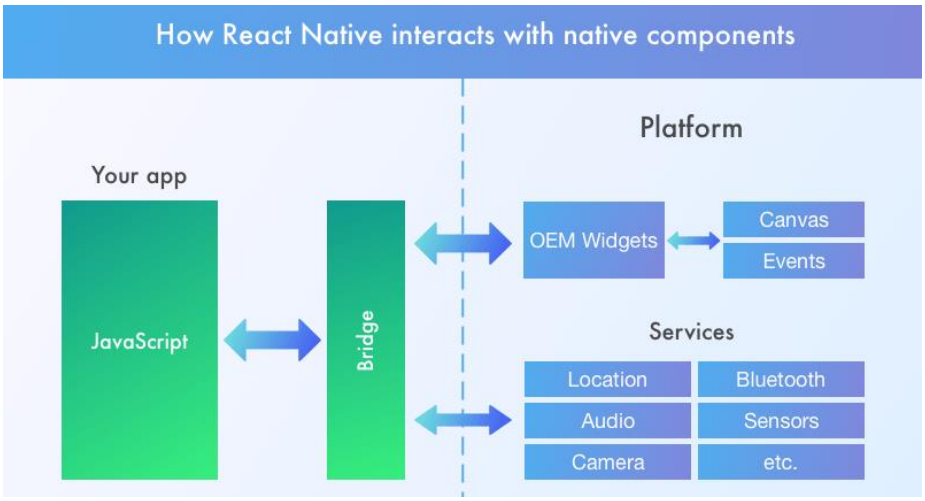

Рисунок 15 – Принцип роботи React Native

Хоча React Native з’явився у 2015 році, але він продовжує активно розвиватися й покращуватися. Завдяки мові Java Script він отримав велику популярність у веб розробників, які отримали можливість створювати мобільні додатки.

2.2.3 Xamarin

Xamarin – найстарший інструмент з розглянутих, він існує майже з 2013 року. Створений однойменною компанією, а згодом придбаний Microsoft. За мову програмування використовує C#, що дозволяє йому працювати швидшу у порівнянні з іншими інструментами, які використовують Java Script.

Однією з особливостей платформи є те, що для кожної операційної системи необхідно створювати інтерфейс використовуючи для цього елементи нативної розробки. Це дозволяє створювати програми, інтерфейс яких максимально наближений до нативних, але вимагає від розробника навичок та додаткового часу.

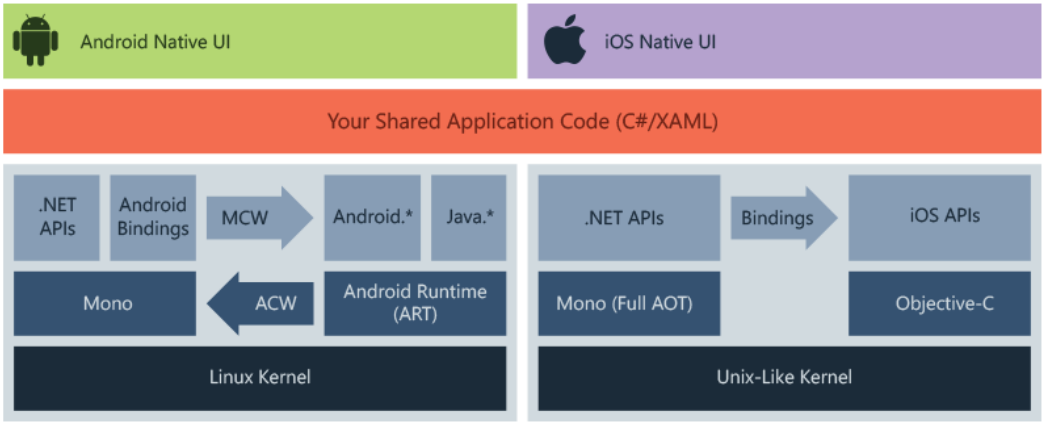

Рисунок 16 – Принцип роботи Xamarin

Для усунення недоліку з інтерфейсом було створено Xamarin Forms. Він додає нові інструменти, що дозволяє створювати один UI для всіх операційних систем.

Xamarin менш популярний у порівняні з React Native та Flutter. Він продовжує свій розвиток, але повільніше за конкурентів. Це пов’язано в більшій мірі з мовою програмування, що використовується фреймворком. C# не така розповсюджена як Java Script, тому робота з Xamarin вимагає вивчення мови програмування.

2.2.4 Обґрунтування вибору фреймворку Flutter

Основним інструментом створення додатку обираємо Flutter, тому що він має багато переваг у порівняні з нативною розробкою або конкурентами іншими кросплатформними інструментами.

Незалежно від того, чи потрібен Android, iOS або веб-програма, Flutter виконує роботу лише за допомогою однієї кодової бази. Звичайно, це те, що повинні робити всі крос-платформні фреймворки, але Flutter робить процес розробки надзвичайно простим, а результати можна спостерігати одразу.

Flutter має зручну функцію під назвою Hot Reload, що безсумнівно, є чудовим інструментом для розробки, збереження часу і полегшує процес створення інтерфейсу.

По суті, Hot Reload – це функція, яка дозволяє розробникам вносити зміни в код і спостерігати за тим, як вони вступають в дію в режимі реального часу. Не потрібно перезавантажувати всю програму, щоб побачити внесені зміни, а у деяких випадках перекомпіляція всіє програми може займати багато часу та потребує потужної робочої станції

Вибравши Flutter, можна розвивати додаток швидше, ніж стандартний нативний додаток. Можна економите час, оскільки створюється лише одна кодову базу для роботи на всіх платформах, а також заощаджується час протягом циклу розробки.

Hot Reload пришвидшує розробку, оскільки ви витрачаєте менше часу на перевірку та зміни. Оскільки є лише одна кодова база для тестування, тестування та налагодження займає менше часу.

Більше того, з Dart , що є мовою програмування Flutter – приємно працювати. Немає класів в інших класах, просте управління доступом, прості числові переліки, ключові слова async / await та ефективне форматування коду − все це робить розробку додатків за допомогою Flutter швидшою, чистою та стислою.

Оскільки додаток Flutter використовує одну і ту ж базу коду на різних платформах, розгортання оновлень – ще одна особливість, простішою та простішою. Можна випускати оновлення як для iOS, так і для Android додатків одночасно, значно зменшуючи зміни в серверній інфраструктурі, а також проблеми впровадження та синхронізації, які часто виникають, коли різні команди розробників працюють на різних платформах.

Після опублікування додатку, можна вкласти ресурси у збільшення бази користувачів та вдосконалення функцій. Обслуговуванням додатків та виправлення помилок, потребує менше коштів, а ніж нативна розробка.

Найбільша особливість, яка відокремлює Flutter від більшості кросплатформних фреймворків – це здатність використовувати функції пристрою для розширення функцій. Flutter може використовувати такі можливості пристрою, як камера, GPS, датчики руху, для забезпечення найкращого користувацького досвіду, а велика кількість готових рішень, що створені спільною, робить використання системних компонентів простим.

За основне середовище розробки будемо використовувати Android Studio. Це інтелектуальне середовище має офіційні доповнення для розробки Flutter проектів. Також, ця програма має вбудований емулятор для запуску на Android пристроях, що виключає встановлення додаткових інструментів. Android Studio включає функції по роботі з Git репозиторіями, тому не потрібно використовувати консольні команди для більшості дій.

Базою для зберігання інформації буде виступати програма WordPress. WordPress – це безкоштовна система управління контентом із відкритим кодом (CMS), написана на PHP і працює в парі з MySQL. Ця програма – одне з найпопулярніших системних рішень, що використовуються, станом на 2021 рік вона використовується більше ніж на 40% сайтів.

WordPress має підтримку сторонніх плагінів, що дозволяє використовувати програми для різних потреб. У нашому випадку, WordPress є

зручним рішення, тому що він має зручний редактор сторінок, який дозволяє у реальному часі редагувати контент використовуючи для цього html.

Використовуючи сторонні плагіни можна додати підтримку REST запитів у систему WordPress. За допомогою них, можна отримувати данні у вигляді html сторінок, які зручно зберігати у локальній базі для можливості офлайн доступу.

3. ПРАКТИЧНА РЕАЛІЗАЦІЯ МОБІЛЬНОГО ДОДАТКУ

Розробка мобільного додатку була розбита на наступні етапи

1. Створення проекту у програмному середовищі Android Studio
2. Створення дизайну та інтерактивного прототипу мобільного додатку
3. Реалізація дизайну безпосередньо у проекті
4. Налаштування системи WordPress на локальному сервері
5. Підключення мобільного додатку до програмного забезпечення WordPress, реалізація запитів до сервера
6. Інтеграція локальної бази даних Hive для можливості офлайн доступу до програми
7. Тестування мобільного додатку
8. Огляд перспективи розвитку додатку

3.1 Опис функціональності

При запуску додатку, першим відкривається Splash Screen (див. рис. 17). Це вікно необхідне для перевірки інтернет з’єднання, завантаження інформації та формування локальної бази даних для офлайн доступу.

Після того, як всі дані завантажено до програми відкривається головне вікно (див. рис. 18). Воно має чотири кнопки, а саме «Каталог препаратів», «Насіння», «Новини», «Контакти». Знизу розташований блок з інформацією про поточну погоду за допомогою геолокації користувача. Якщо доступ до місцезнаходження заборонено то відображається погода у місті Києві.

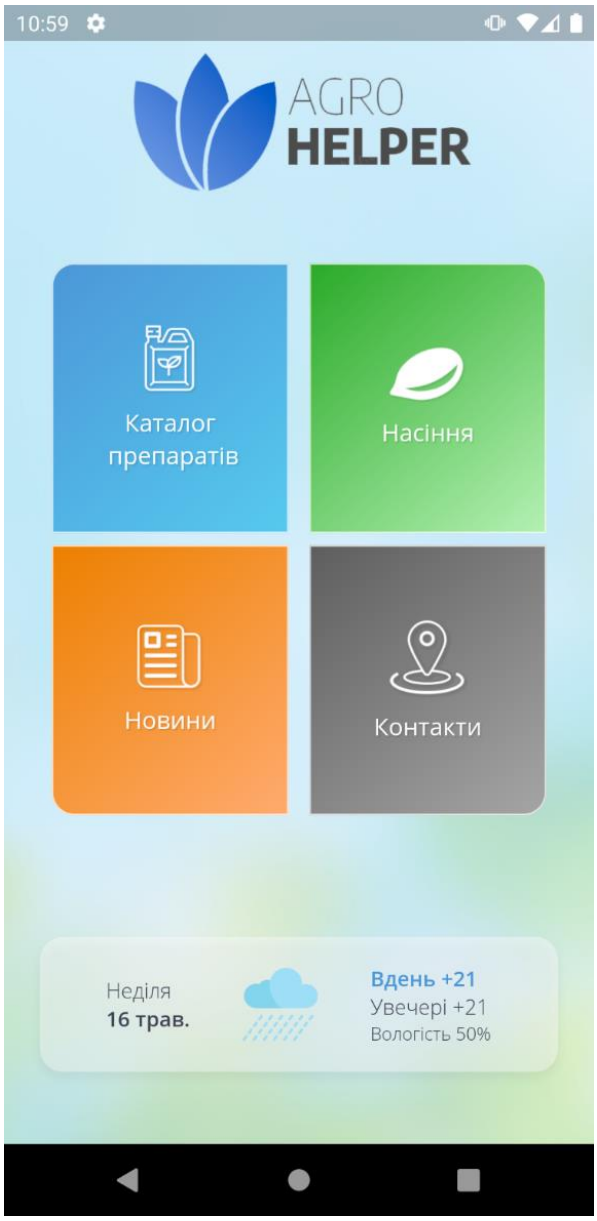

Рисунок 17 – Головне вікно

Рисунок 18 – Splash Screen

При натисканні на кнопку «Каталог препаратів» з’являється екран з категоріями (див. рис. 19). При натисканні на категорію відкривається список препаратів що входять до неї. У верхній частині екрану розташована кнопка пошуку та кнопка переходу на попередній екран.

Рисунок 19 – Каталог препаратів

Вікно пошуку має поле для вводу запиту та кнопку для здійснення пошуку (див. рис. 20). Пошук відбувається одразу по всім категоріям. Під час запиту враховуєтеся не лише назва препарату, а й інформація про нього, даючи можливість шукати препарати по активним речовинам, або способу застосування.

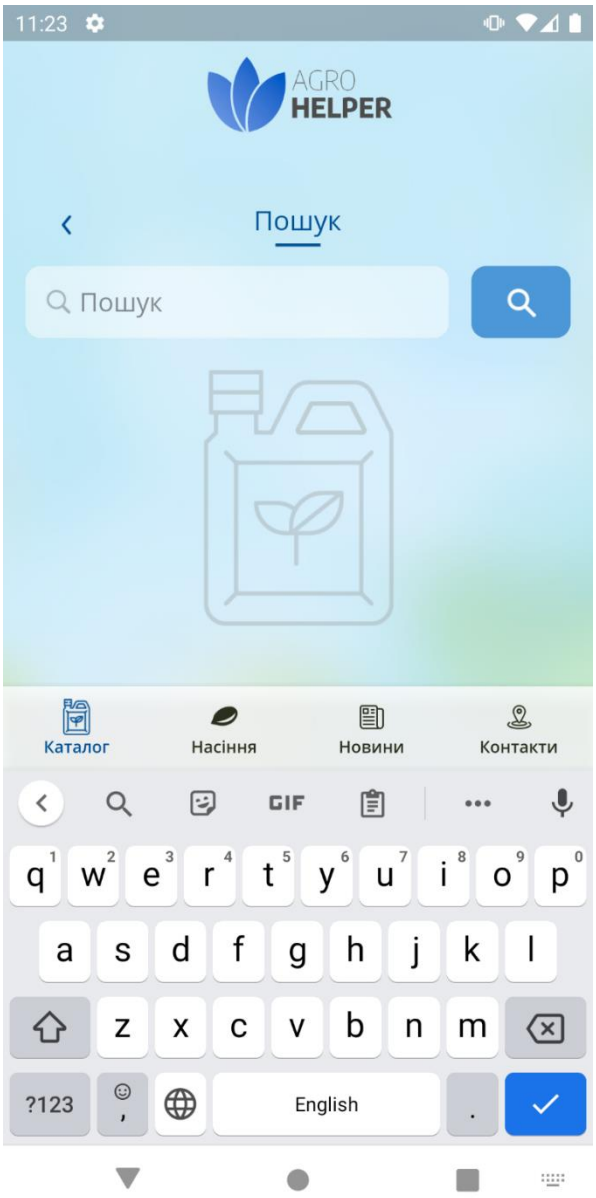

Рисунок 20 – Вікно пошуку

Екран зі препаратами має список, що складається з їхніх назв (див. рис. 21). У верхній частині знаходиться кнопка фільтру. При натисканні на кожний з них відображається вікно з детальною інформацією (див. рис. 22). Воно може мати не лише простий текст, а й таблиці або зображення.

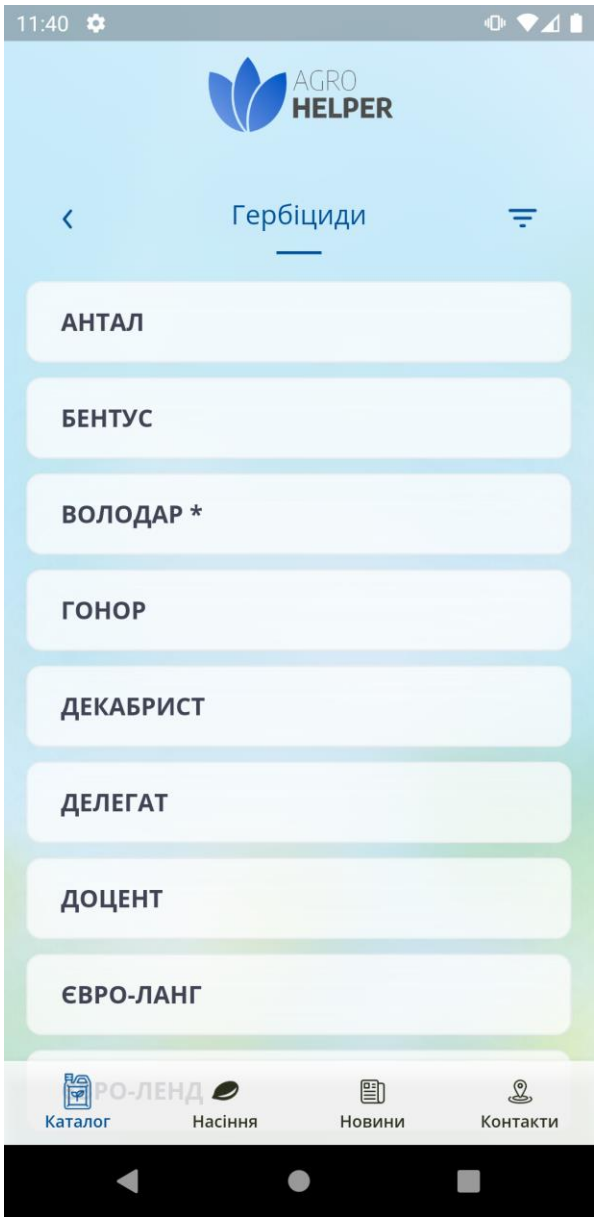

Рисунок 21 – Список препаратів

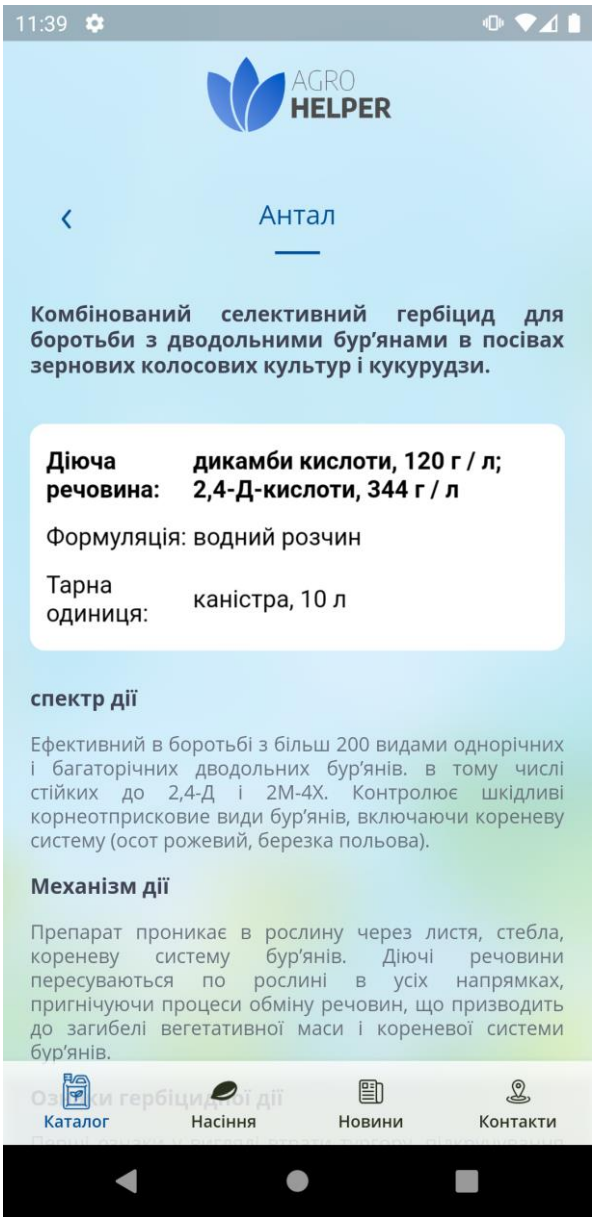

Рисунок 22 – Детальна інформація про препарат

Вікно фільтрації складається з двох категорій до яких можна застосувати фільтр – це «Оброблювальні культури» та «» Діючі речовини» (див. рис. 23). Вони виконують фільтр за культурами до яких можна застосовувати препарат та діючі речовини які входять до його складу. При натисканні на категорії знизу з'являється діалогове вікно зі списком можливих варіантів (див. рис. 24). Фільтр застосовується при натисканні на кнопку «Застосувати» у нижній частині.

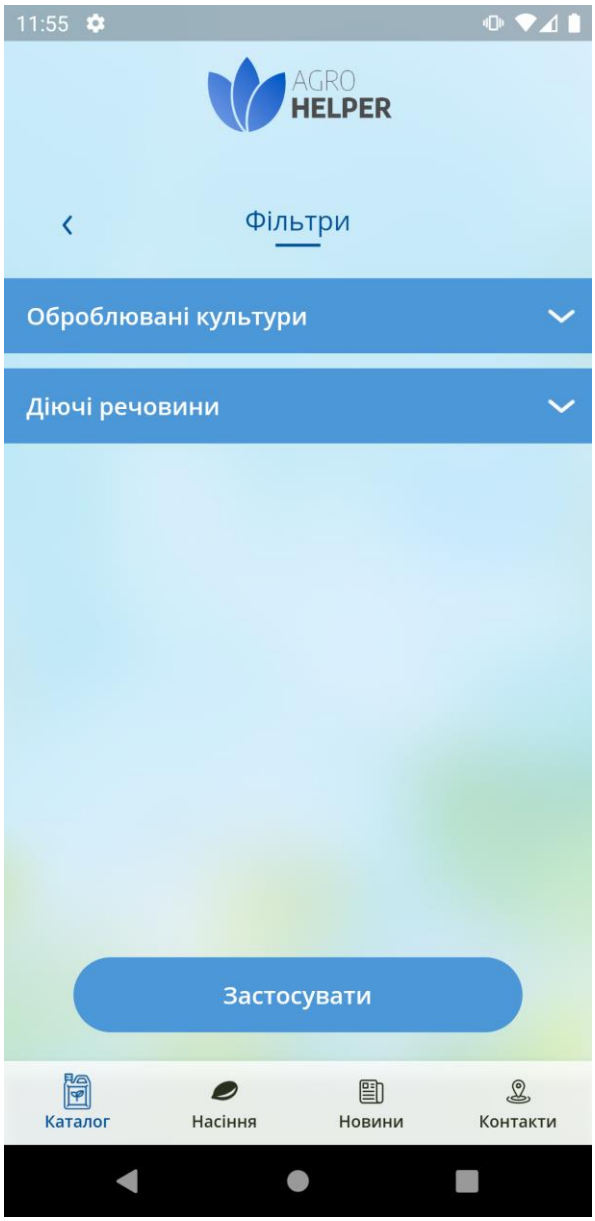

Рисунок 23 – Вікно з фільтром

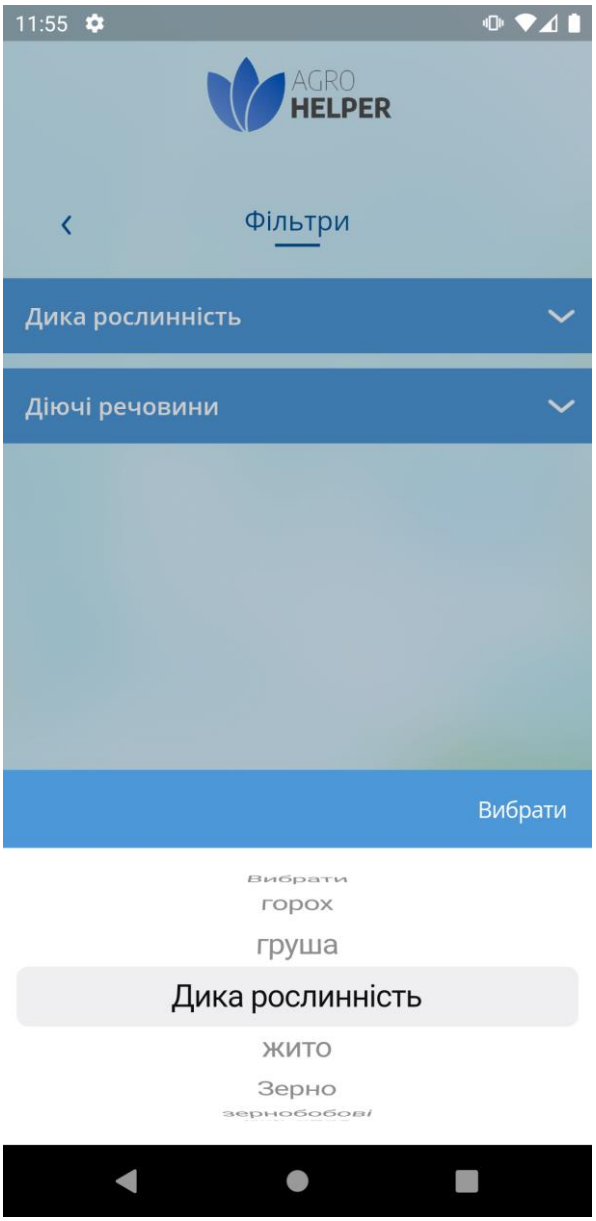

Рисунок 24 – Діалогове вікно зі списком варіантів

Вкладка з типами насіння має список з різними культурами (див. рис. 25). Кожний елемент списку має свою назву, короткий опис та зображення для легкої та інтуїтивної навігації. При кліку на елемент відкривається список гібридів культури.

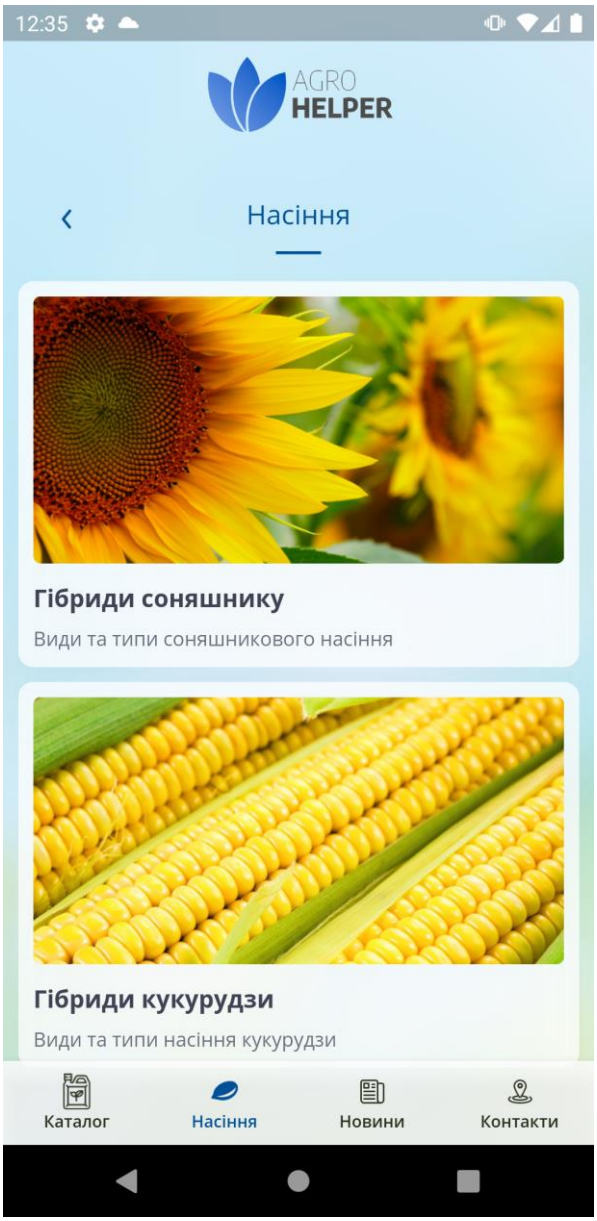

Рисунок 25 – Вікно з насінням

Кожний елемент списку гібридів насіння має свою назву та додаткові відомості (див. рис. 26), при натисканні відкривається сторінка з більш детальною інформацією (див. рис. 27). Вона може включати текст, таблиці та зображення гібриду.

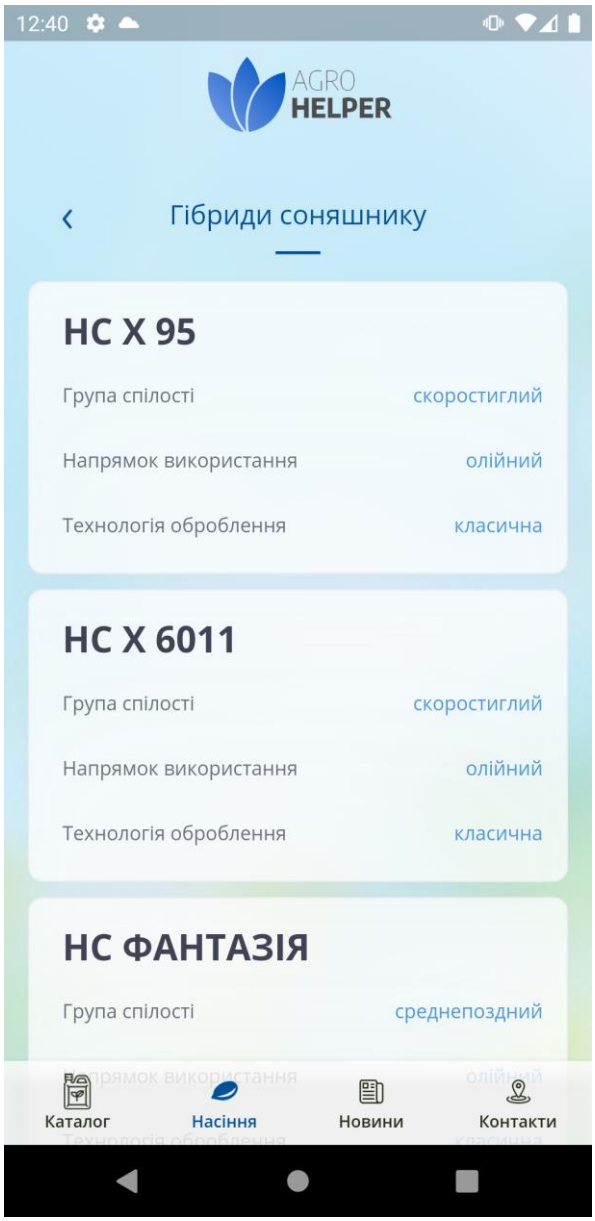

Рисунок 26 – Список гібридів

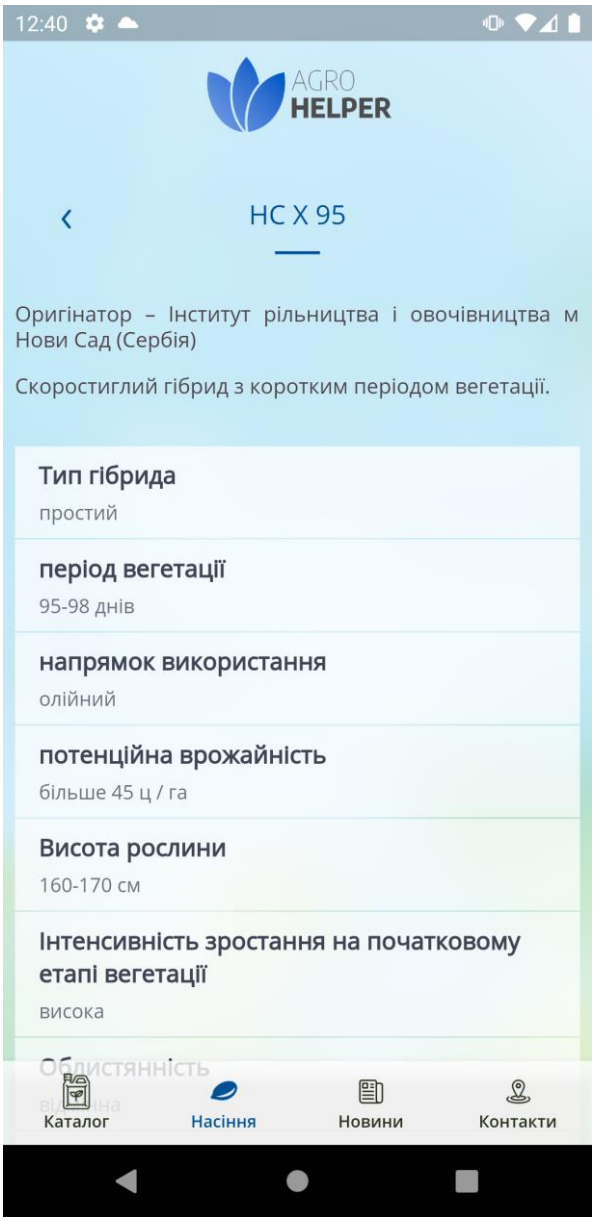

Рисунок 27 – Детальна інформація про гібрид

Сторінка з новинами містить список з актуальними подіями, що відбуваються у агросфері країни (див. рис. 28). Новини оновлюються автоматично та не доступні у офлайн режимі. Елемент списку містить заголовок, короткий опис та дату публікації. При натисканні відкривається вікно з повним вмістом події (див. рис. 29). Воно може включати зображення, текст, посилання або відео.

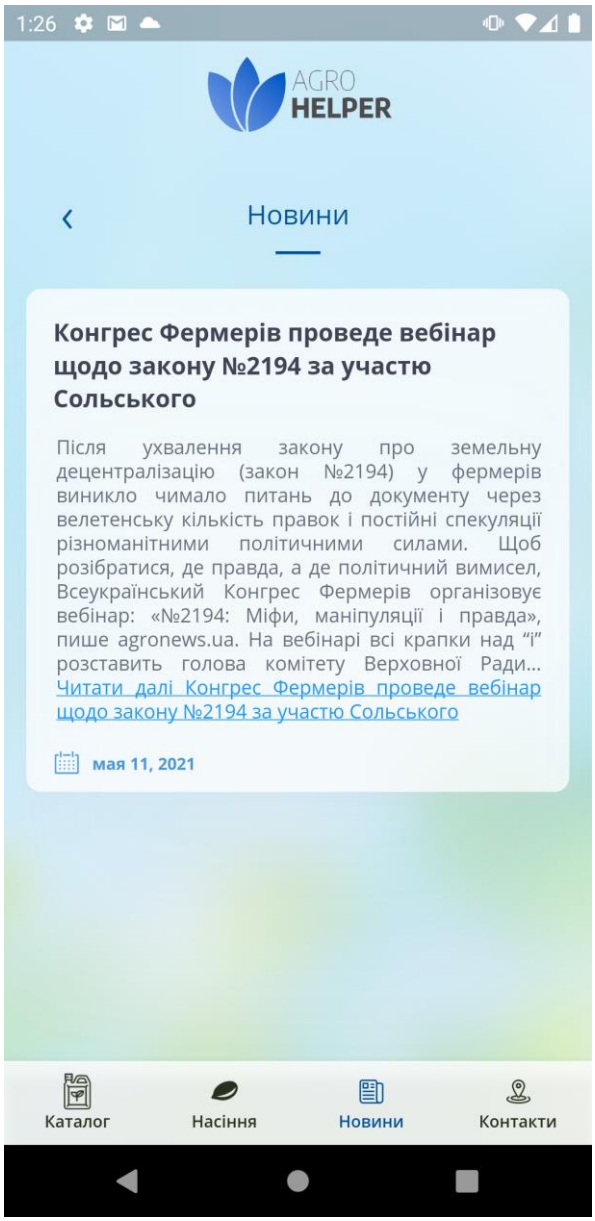

Рисунок 28 – Список новин

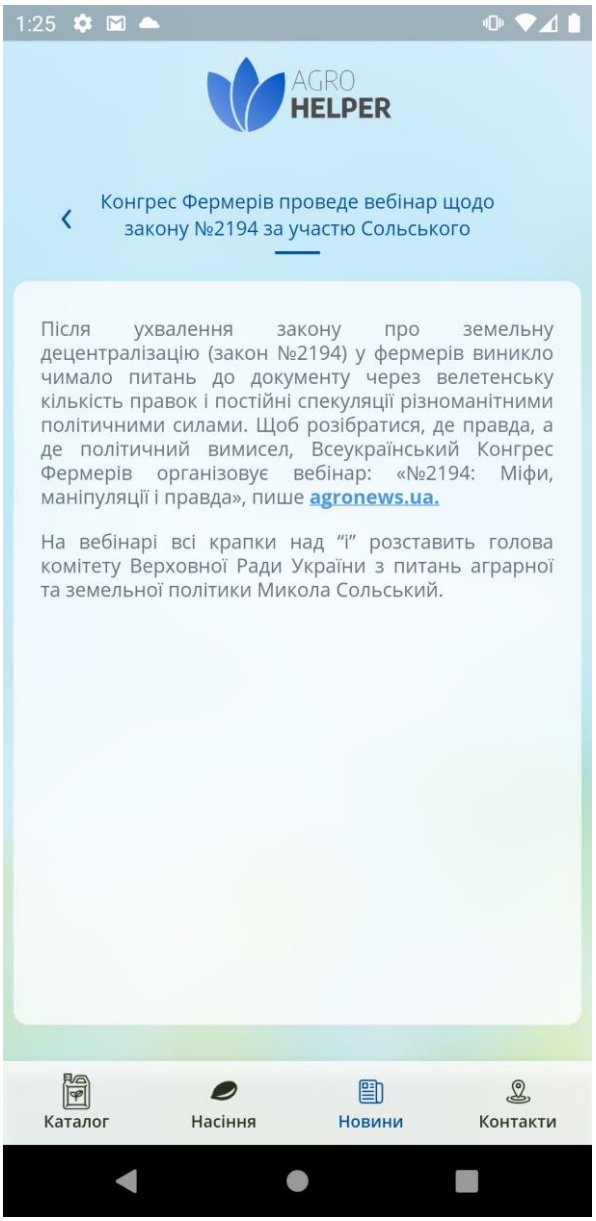

Рисунок 29 – Екран детального огляду новини

Екран контактів складається зі списку фірм виробників, або дистриб'юторів закордонної продукції (див. рис. 30). При натисканні на елемент списку, відкривається нове вікно з адресом, номером телефону та email, в різних містах України (див. рис. 31). Для зручності користувача присутні Google Maps, на яких відображається місцезнаходження офісу. Також, користувач має можливість зателефонувати або написати прямо з додатку використовуючи відповідні кнопки.

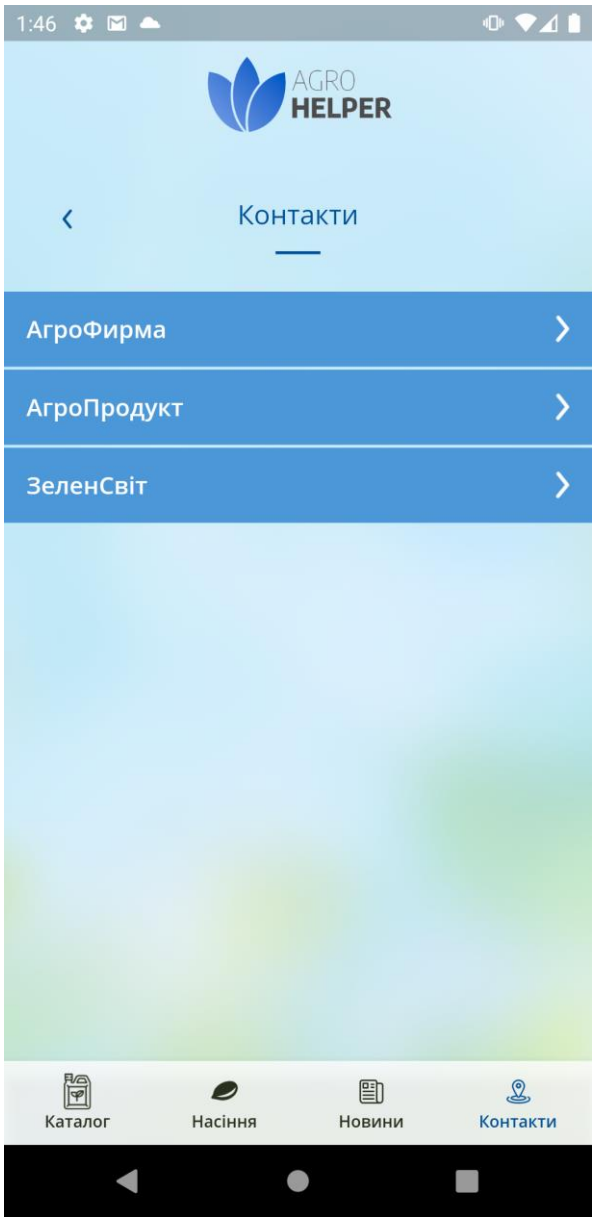

Рисунок 30 – Список контактів

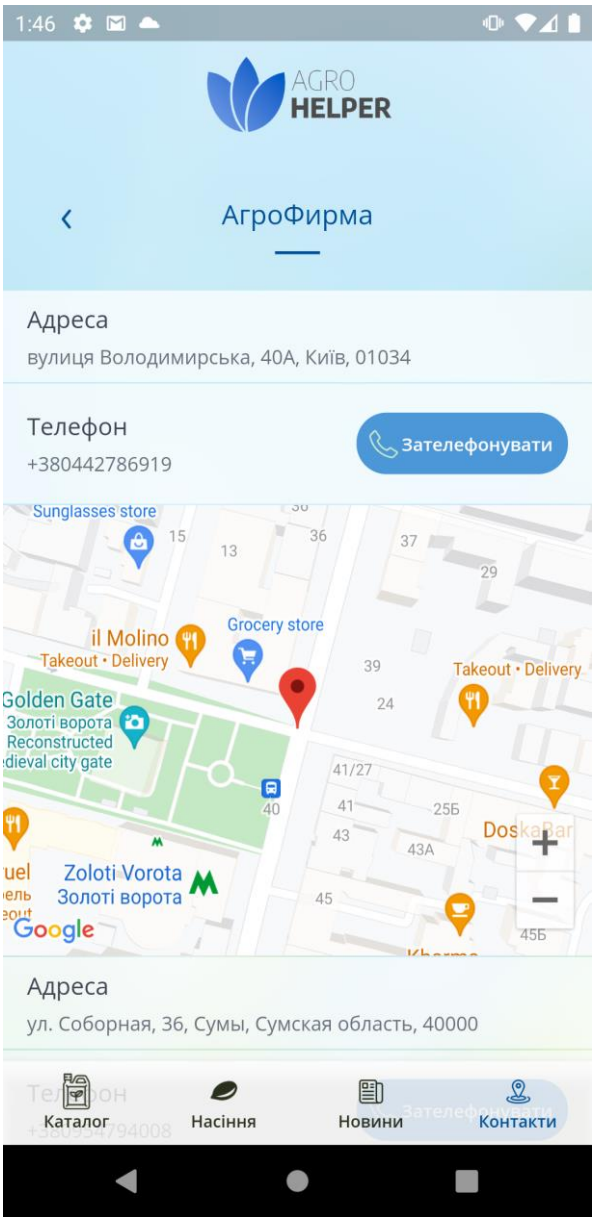

Рисунок 31 – Детальний огляд контактної інформації

3.2 Архітектура додатку

Архітектура програмного забезпечення – це одна з найважливіших речей під час проектування програми. Вона створює фундамент до якого потім додаються сервіси та користувацький інтерфейс. Також, від архітектури залежить наскільки легко підтримувати додаток у майбутньому, як додавати нові функції щоб не вплинути на роботу всього програмного коду. Якщо

знехтувати архітектурою й почати створення програми без неї то такий код зі збільшенням об’єму стає складнішим для розуміння.

Під час створення додатку було обрано архітектурний патерн Stacked. Stacked – це одна з варіацій архітектури MVVM. Основна логіка патерну складається з трьох основних частин:

● View відображає інтерфейс для користувача. Окремі віджети також кваліфікуються як view, в даному випадку, не є екраном, а лише окремим елементом інтерфейсу.

● ViewModel керує станом View, бізнес-логікою та будь-якою іншою логікою, в залежності від дій користувача. ViewModel зазвичай використовує сервіси.

● Services – обгортка над набором певних функцій. Сервіс зазвичай використовується для таких дій, як показ діалогового вікна, робота з базою даних, інтеграція API тощо. Один сервіс можна використовувати одразу для декількох екранів, що зменшує кількість дублюючого коду.

При використанні цього патерну, також потрібно дотримуватися деяких правил. Не дотримання може привести до не правильної роботи коду та виникненню помилок. Деякі з цих правил наступні:

● View ніколи не повинні безпосередньо використовувати сервіси. Всі взаємодії повинні відбуватися через ViewModel.

● View не повинні містити логіку. Якщо логіка належить лише до елементів інтерфейсу, то її можна лишити, але решту слід передати до ViewModel.

● View має оновлювати свій стан лише використовуючи свою ViewModel.

● ViewModels можуть бути використані повторно, якщо користувальницький інтерфейс двох View вимагає схожих функцій.

● ViewModel не повинна знати про інші ViewModel.

У реалізації шаблону Stacked допомагає однойменний пакет, який містить у собі необхідний набір функцій, що прискорює використання патерну.

У проекті також було використано впровадження залежності. Впровадження — це передача залежності залежному об'єкту. Основна суть цього підходу є передача залежності об'єкту замість створення нового сервісу всередині об’єкту, що є базовою вимогою для використання цього шаблону.

Впровадження залежності зручний інструмент, який допомагає у менеджменті залежностей та створенні класів. Він гарно компонує з патерном Stacked. Зникає необхідність у створенні нових об’єктів сервісів для передачі їх у ViewModel. Основні переваги використання впровадження:

● Введення залежності дозволяє об’єкту позбавитись знання про конкретну реалізацію, яку він повинен використовувати. Це допомагає ізолювати об’єкт від впливу змін конструкції.

● Введення залежності дозволяє налагодити одночасну та незалежну розробку. Декілька розробників можуть самостійно розробляти класи, які використовують один одного, при цьому достатньо знати інтерфейс, через який будуть спілкуватися класи.

● Введення залежності зменшує зв'язок між об’єктом та його залежністю.

● Впровадження залежностей можна використовувати для конфігурації системи без її перекомпіляції. Для цього створюють різні конфігураційні файли для різних ситуацій.

Для обміну інформацією між сервером та додатком використовується REST API. Реалізацію цього архітектурного стилю забезпечує система WordPress. Вона здатна приймати звичайні HTTP запити такі як POST або GET та надає повний доступ для роботи з базою даних в якій зберігаються сторінки.

Дані передаються за допомогою JSON – це відкритий формат, що використовується для збереження або передачі даних [10]. Спочатку

використовувався як компонент мови JavaScript, але з часом більшість мов отримали підтримку цього формату. Це зручний для розуміння і використання формат, його легко розпарсити на програмні класи і передавати інформацію між екранами. Також, JSON зручно зберігати у локальній базі на пристрої користувача.

Локальна база даних реалізована з використанням пакету Hive. Це Non-SQL база даних що зберігає дані у форматі JSON. Використання Hive зумовлено гарною швидкістю роботи в порівняні з конкурентами, простотою використання, можливість використовувати вбудоване шифрування для забезпечення безпеки конфіденційних даних.

Форматування сторінок препаратів, насіння та новин відбувається з використанням HTML розмітки. HTML теги розпізнаються програмою, після до вмісту тегу застосовується певний стиль форматування. У випадку зі складними конструкціями, такі як таблиці, що мають більше 3 колонок використовується web view. Програма попри звичайного тексту та таблиць підтримує посилання, відео та зображення.

Підхід з використанням HTML розмітки дозволяє форматувати сторінки в залежності від потреб та не прив’язуватись до певного шаблону. Це зручно, адже різні препаратів можуть відрізнятися за кількістю інформації або властивостей.

Основні файли проекту розташовані у окремих директоріях (див. рис.32). Це зроблено для зручності навігації по проекту.

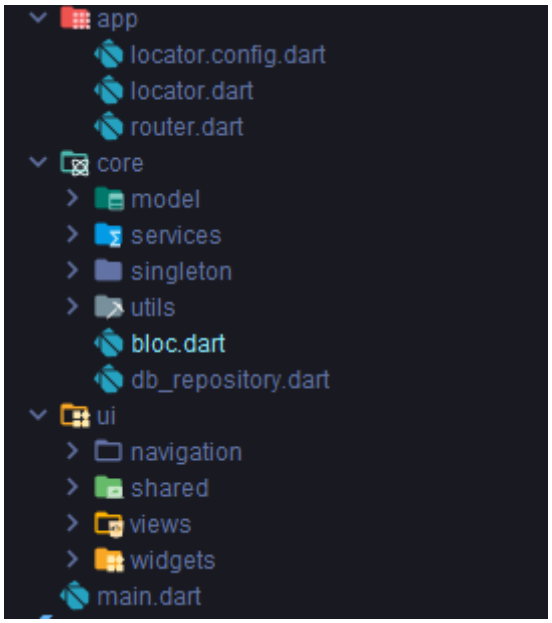

Рисунок 32 – Структура проекту

У папці app містяться файли що відповідають за навігацію по сторінкам та за реалізацію впровадження залежностей.

Папка core містить файли моделей, сервіси та утиліти. Моделі використовуються для парсингу JSON формату у програмні класи для подальшого використання. Сервіси надають доступ до бази даних, отримання інформації з WordPress та відповідають за погоду. Утиліти допомагають виконувати дії, що часто виконуються в коді, більшість цих методів статичні, тому їх виконання не вимагає створення нового об’єкту.

Директорія ui включає візуальні представлення. До її складу входять View, що представляють собою певний екран та ViewModel, що містить логіку для нього. Також, тут розташовані кастомні віджети, вони створені для зменшення коду інтерфейсу у view. Основні кольори, що використовуються у програмі також винесені у окремий файл для зручності кастомізації.

Файл main.dart є точкою входу, з нього починається запуск та виконання програми. Ініціалізація локальної бази даних, мови додатку та основних налаштування відбувається у цьому файлі.

3.3 Тестування додатку

Тестування програмного забезпечення − це організаційний процес розробки програмного забезпечення, в рамках якого критично важливе для бізнесу програмне забезпечення перевіряється на правильність, якість та ефективність.

Тестування програмного забезпечення дозволить заощадити час та гроші організації, зменшивши витрати на розробку та обслуговування програмного забезпечення. Тестування створює гарантії стабільності при розробці нових функцій та гарантує що вона буде працює належним чином, і користувачі не зіткнуться з помилками.

У процесі розробки програми проводилося поетапне тестування з метою виявлення програмних помилок і невідповідні з технічним завданням.

Для цього були створені емулятори смартфона і планшета з різними діагоналями екрана для різних версій Android та iOS. Програмний продукт послідовно запускався на цих емуляторах, його поведінка аналізувалося, і при необхідності по результатами аналізу вносилися зміни в код.

Під час тестування роботи додатку з сервером в код були додані спеціальні функції, для виявлення помилок при надсиланні запитів та отримання відповідей, у випадку помилки, до системного журналу виводилась необхідна інформація для виправлення проблеми. Наприклад, при отриманні відповіді від сервера перевірявся код, з яким сервер відповів на запит. У разі 200-го коду тест пройдено в іншому разі виконання програми зупинялось, а детальна інформація про помилку друкувалась до системного журналу.

Були проведені наведені нижче тести

1. Додаток було запущено на пристроях, що працюють під керуванням різних версій Android та iOS з метою виявлення особливостей роботи програми, запущеного в різних версіях операційної системи.

2. Програма тестувалася на пристроях з різної діагоналлю, починаючи від смартфонів з невеликими екранами розміром в 4.7 дюйма та закінчуючи 12 дюймовими планшетами.

3. Після завершення циклу розробки, програмний продукт тестувався на реальних пристроях.

4. До програми додано систему Firebase crashlytics для трекінгу помилок на пристроях користувачів та їх оперативного усунення.

3.4 Перспективи розвитку мобільного додатку

Розвиток мобільного додатку та пост релізна підтримка допомагають утримувати існуючих користувачів та отримати нову аудиторію, тому необхідно запроваджувати нові функції та виправляти помилки.

Програма має значні можливості для розширення свого функціоналу. Завдяки використанню крос платформного фреймворку вона має єдину кодову базу, завдяки якій інтеграція нових можливостей та виправлення помилок займає вдвічі менше часу та ресурсів.

Першою можливою функцією для впровадження є калькулятор сумісності препаратів. Оскільки, не всі препарати сумісні між собою, така функція буде корисною для користувачів. Цей калькулятор буде показувати які препарати можна міксувати, а які є несумісними між собою. Також, він покаже у якій послідовності потрібно змішувати хімікати.

З цим калькулятором зникає необхідні у пошуку інформації на сторінці кожного препарату, користувач зможе обрати їх зі списку, після того програма видає результат.

Наступна функція створення системи для оновлення локальної бази на пристрої користувача. Зараз це відбувається кожного разу, при наявності Інтернету, але в цьому не має необхідності, адже база даних не оновлюється з такою високою періодичністю. Такий підхід лише витрачає інтернет трафік.

Система оновлень буде отримувати поточну версію бази з серверу и якщо вона є вищою ніж локальна то користувачу виводиться діалогове вікно з можливістю оновлення. Він може погодитися, або відкласти це оновлення на потім. Якщо пристрій не має доступу до мережі то автоматично завантажуєтеся локальна версія.

З такою системою, програма буде швидше завантажуватися адже не потрібно завантажувати всі дані кожний раз та економити об’єм трафіку.

Для зручності користування до програми можна додати підтримку декількох мов, як наприклад російської, української та англійської. Декілька мов дозволять користувачам обирати ту, яка найбільше зручна для них. Підтримка англійської мови також дозволяє додатку вийти на міжнародній ринок, або адаптувати його під конкретну країну.

Важливо підтримувати зв’язок з аудиторією, адже саме вона може пропонувати нові можливості що дійсно необхідні. На основі відгуків та оцінок можна виділити інші необхідні для користувачів функції та поступово впроваджувати їх у програму.

ВИСНОВКИ

У ході виконання роботи було проаналізовано наявні на ринку додатки, що відповідають тематиці інформаційної системи для агробізнесу, а саме «Сингента Украина», «Corteva Agriscience UA» та «Summit-Agro Ukraine».

У результаті аналізу були висунуті основні вимоги та реалізовано мобільний додаток за допомогою кросплатформного інструменту Flutter.

Додаток має наступні функції:

- Перегляд необхідної інформації про препарати
- Доступ до додатку офлайн
- Автоматичне оновлення бази даних при підключенні до інтернету
- Пошук та фільтр для зручного сортування інформації
- Відображення інформації про погоду за допомогою геолокації користувача
- Вкладка з новинами в агросфері
- Відомості про насіння та їх гібриди
- Список з контактами агрофірм та дистриб’юторів
- Додаток протестовано на різних версіях операційних систем Android та iOS та різних діагоналях.

Також, зважаючи на стрімкий розвиток агробізнесу в Україні, слід виділити актуальність розробки даного програмного забезпечення з використанням останніх технологій.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Dawn Griffiths; Head First Android Development: A Brain-Friendly Guide. — O'Reilly Media, 2017. — 887 с. — ISBN 9781491974056.
2. Герберт Шілдт: Java. The Complete Reference, 10th Edition. — М .: «Діалектика», 2018. — 1488 с. — ISBN 978-5-6040043-6-4.
3. Hagos, Ted: Android Studio IDE Quick Reference. A Pocket Guide to Android Studio Development. — Apress, 2019. —182 с. — ISBN 978-1-4842-4952-9.
4. Hillegass, Aaron; Conway, Jon: iOS Programming: The Big Nerd Ranch Guide (3rd ed.). — Pearson, 2012. — ISBN 978-0-321-82152-2.
5. Swift.org: About Swift [Електронний ресурс] – Режим доступу до ресурсу: https://docs.swift.org/swift-book/
6. Knott, Matthew: Beginning Xcode. Swift 3 Edition. — Apress, 2016. — 463 с. — ISBN 978-1-4302-5004-3.
7. Habr.com: Про Flutter, кратко: Основы [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/post/430918/.
8. Walrath, K. and Ladd, S. Dart: Up and Running. — O'Reilly, 2012. — 152 p. — ISBN 9781449330897.
9. Flutter.dev: Developing packages & plugins [Електронний ресурс] – Режим доступу до ресурс: https://flutter.dev/docs/development/packages-and-plugins/developing-packages.
10. Ecma-international.org: ECMA-404 [Електронний ресурс] – Режим доступу до ресурс: https://www.ecma-international.org/publications-and-standards/standards/ecma-404/.

ДОДАТОК А

Додаток А1. Лістинг коду головного файлу main.dart

```
import 'package:agrohelper/app/router.dart';
import 'package:agrohelper/core/model/corn_model.dart';
import 'package:agrohelper/core/model/culture_model.dart';
import 'package:agrohelper/core/model/default_model.dart';
import 'package:agrohelper/core/model/sunflower_model.dart';
import 'package:agrohelper/core/model/topic_model.dart';
import 'package:agrohelper/ui/shared/my_colors.dart';
import 'package:easy_localization/easy_localization.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:hive/hive.dart';
import 'package:injectable/injectable.dart';
import 'package:path_provider/path_provider.dart';

import 'app/locator.dart';

Future<void> main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await EasyLocalization.ensureInitialized();
  await configureDependencies(Environment.prod);
  var dir = await getApplicationDocumentsDirectory();
  Hive
    ..init(dir.path)
    ..registerAdapter(DefaultModelAdapter())
    ..registerAdapter(CultureModelAdapter())
    ..registerAdapter(SunflowerModelAdapter())
    ..registerAdapter(SunflowerBoxAdapter())
    ..registerAdapter(DefaultBoxAdapter())
    ..registerAdapter(CornBoxAdapter())
    ..registerAdapter(TopicBoxAdapter())
    ..registerAdapter(CultureBoxAdapter())
    ..registerAdapter(CornModelAdapter())
    ..registerAdapter(TopicModelAdapter());
  runApp(EasyLocalization(
      path: 'assets/translations',
      fallbackLocale: Locale("uk"),
      supportedLocales: [Locale("uk")],
      child: MyApp()));
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return GetMaterialApp(
      title: 'ArgoHelper',
      debugShowCheckedModeBanner: false,
      localizationsDelegates: context.localizationDelegates,
      supportedLocales: context.supportedLocales,
      locale: context.locale,
      getPages: getPages,
      theme: ThemeData(
        accentColor: AppColors.blue,
        textButtonTheme: TextButtonThemeData(
```

```
        style: TextButton.styleFrom(          primary: AppColors.darkBlue
        )
      ),
      primarySwatch: AppColors.generateMaterialColor(AppColors.blue),
    ),
    initialRoute: initialRoute,
  );
 }
}
```

Додаток А2. Лістинг коду головної сторінки

```
import 'package:agrohelper/app/locator.dart';
import 'package:agrohelper/core/model/weather_model.dart';
import 'package:agrohelper/ui/shared/my_colors.dart';
import 'package:agrohelper/ui/views/home_view/home_view_model.dart';
import 'package:agrohelper/ui/views/main_view/main_view_model.dart';
import 'package:agrohelper/ui/widgets/main_item_view.dart';
import 'package:agrohelper/ui/widgets/weather_widget.dart';
import 'package:easy_localization/easy_localization.dart';
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:stacked/stacked.dart';

class MainView extends StatefulWidget {
 @override
 _MainViewState createState() => _MainViewState();
}

class _MainViewState extends State<MainView> {
 @override
 Widget build(BuildContext context) {
  return ViewModelBuilder<MainViewModel>.reactive(
    viewModelBuilder: () => locator<MainViewModel>(),
    onModelReady: (model) => model.init(),
    builder: (context, model, widget) {
     return Scaffold(
      key: model.scaffoldKey,
      body: Stack(
       fit: StackFit.expand,
       children: <Widget>[
        Image.asset(
         "assets/background.png",
         fit: BoxFit.cover,
        ),
        SafeArea(
         child: Padding(
          padding: EdgeInsets.only(left: 24, right: 24),
          child: Column(
           mainAxisSize: MainAxisSize.max,
           mainAxisAlignment: MainAxisAlignment.spaceBetween,
           children: <Widget>[
            Container(
             width: MediaQuery.of(context).size.width,
             height: 115,
             child: Stack(
```

```
    children: <Widget>[
     Align(
      child: Image.asset(
       "assets/logo.png",
       fit: BoxFit.cover,
       height: 115,
       width: 230,
      ),
      alignment: Alignment.center,
     ),
    ],
   ),
  ),
  Flexible(
   fit: FlexFit.loose,
   child: ConstrainedBox(
    constraints: BoxConstraints(
     maxHeight: 368,
     maxWidth: 329,
    ),
    child: Column(
     children: <Widget>[
      Expanded(
       child: Row(
        mainAxisSize: MainAxisSize.min,
        children: <Widget>[
         MainItemButton(
          maxWidth: 36,
          iconPath: "assets/canister.png",
          borderRadius: BorderRadius.only(
            topLeft: Radius.circular(15)),
          title: tr("preparat"),
          colors: [
           Color(0xFF4C97D8),
           Color(0xFF56CAEF)
          ],
          onPressed: () => {
           model.openHomeScreen(TabItem.catalog)
          },
         ),
         SizedBox(width: 14),
         MainItemButton(
          maxWidth: 52,
          iconPath: "assets/seeds.png",
          borderRadius: BorderRadius.only(
            topRight: Radius.circular(15)),
          title: tr("seeds"),
          colors: [
           Color(0xFF2AAA28),
           Color(0xFFB5F1B4)
          ],
          onPressed: () => {
           model.openHomeScreen(TabItem.seeds)
          },
         ),
        ],
       ),
      ),
```

```
      SizedBox(height: 8),
      Expanded(
       child: Row(
        mainAxisSize: MainAxisSize.min,
        children: <Widget>[
         MainItemButton(
          maxWidth: 45,
          iconPath: "assets/news.png",
          borderRadius: BorderRadius.only(
             bottomLeft: Radius.circular(15)),
          title: tr("news"),
          colors: [
           Color(0xFFED8000),
           Color(0xFFFFAB6F)
          ],
          onPressed: () =>
             model.openHomeScreen(TabItem.news),
         ),
         SizedBox(width: 14),
         MainItemButton(
          maxWidth: 53,
          iconPath: "assets/contacts.png",
          borderRadius: BorderRadius.only(
             bottomRight: Radius.circular(15)),
          title: tr("contacts"),
          colors: [
           Color(0xFF5E5E5E),
           Color(0xFFA6A6A6)
          ],
          onPressed: () => model
             .openHomeScreen(TabItem.contacts),
         ),
        ],
       ),
      )
     ],
    ),
   ),
),
SizedBox(
 height: 15,
),
Container(
   height: 90,
   alignment: Alignment.center,
   decoration: BoxDecoration(
      borderRadius:
         BorderRadius.all(Radius.circular(15)),
      boxShadow: <BoxShadow>[
       BoxShadow(
          color: Color.fromRGBO(0, 0, 0, 0.05),
          blurRadius: 4,
          offset: Offset(0, 4))
      ],
      color: Color.fromRGBO(255, 255, 255, 0.4)),
   child: StreamBuilder<WeatherModel>(
      stream: model.weather,
      builder: (context, snapshot) {
```

```
                        if (snapshot.hasData) {
                          Daily daily = snapshot.data.daily.first;
                          return WeatherWidget(
                            daily: daily,
                          );
                        } else if (snapshot.hasError) {
                          return Icon(
                            Icons.error_outline,
                            size: 32,
                            color: AppColors.blue,
                          );
                        } else {
                          return CircularProgressIndicator();
                        }
                      })),
                  SizedBox(
                    height: 15,
                  )
                ],
              ),
            ),
          ),
        ],
      ),
    );
  });
 }
}
```

```
import 'package:agrohelper/app/router.dart';
import 'package:agrohelper/core/model/weather_model.dart';
import 'package:agrohelper/core/services/navigation/navigation_service.dart';
import 'package:agrohelper/core/services/weather/weather_service.dart';
import 'package:agrohelper/core/utils/constants.dart';
import 'package:agrohelper/ui/views/home_view/home_view_model.dart';
import 'package:flutter/material.dart';
import 'package:geolocator/geolocator.dart';
import 'package:injectable/injectable.dart';
import 'package:rxdart/rxdart.dart';
import 'package:stacked/stacked.dart';
import 'package:url_launcher/url_launcher.dart';

@injectable
class MainViewModel extends BaseViewModel {
 final NavigationService _navigationService;
 final WeatherService _weatherService;

 final _weatherFetcher = PublishSubject<WeatherModel>();

 MainViewModel(this._navigationService, this._weatherService);

 Stream<WeatherModel> get weather => _weatherFetcher.stream;

 final GlobalKey<ScaffoldState> scaffoldKey = GlobalKey();

 void init() {
  getWeather();
 }
```

```
loadDefaultWeather() {
  fetchWeather(Constants.defaultLatitude, Constants.defaultLongitude);
  showSnackBar();
}

getWeather() async {
  var permission = await Geolocator.checkPermission();
  if (permission == LocationPermission.denied) {
    permission = await Geolocator.requestPermission();
    if (permission == LocationPermission.denied) {
      loadDefaultWeather();
      return;
    }
  }

  bool isEnable = await Geolocator.isLocationServiceEnabled();
  if (isEnable) {
    Geolocator.getCurrentPosition(timeLimit: Duration(seconds: 8)).then(
        (position) {
      if (position != null) {
        fetchWeather(position.latitude, position.longitude);
      } else {
        loadDefaultWeather();
      }
    }, onError: (e) {
      print(e);
      loadDefaultWeather();
    }).catchError((e) {
      print(e);
      loadDefaultWeather();
    });
  } else {
    loadDefaultWeather();
  }
}

showSnackBar() {
  final snackBar = SnackBar(
      content:
          Text('Геолокація недоступна, за замовчуванням обрано місто Київ'));
  scaffoldKey.currentState.showSnackBar(snackBar);
}

void openHomeScreen(TabItem tabItem) {
  _navigationService.pushNamed(Routes.HomeView,
      arguments: HomeViewArguments(tabItem: tabItem));
}

launchURL(String url) async {
  if (await canLaunch(url)) {
    await launch(url);
  }
}

fetchWeather(double lat, double lon) async {
  _weatherService.fetchWeather(lat, lon).then((value) {
    _weatherFetcher.sink.add(value);
```

```
    }, onError: (error) {
      _weatherFetcher.sink.addError(error);
    });
  }

  @override
  void dispose() {
    _weatherFetcher.close();
    super.dispose();
  }
}
```

Додаток А3. Лістинг коду домашньої сторінки

```
import 'package:agrohelper/app/locator.dart';
import 'package:agrohelper/ui/navigation/catalog_tab_navigator.dart';
import 'package:agrohelper/ui/navigation/contacts_tab_navigator.dart';
import 'package:agrohelper/ui/navigation/news_tab_navigator.dart';
import 'package:agrohelper/ui/navigation/seeds_tab_navigator.dart';
import 'package:agrohelper/ui/views/home_view/home_view_model.dart';
import 'package:agrohelper/ui/widgets/bottom_navigation.dart';
import 'package:flutter/material.dart';
import 'package:flutter/rendering.dart';
import 'package:stacked/stacked.dart';

class HomeView extends StatefulWidget {
  const HomeView({Key key}) : super(key: key);

  @override
  _HomeViewState createState() => _HomeViewState();
}

class _HomeViewState extends State<HomeView> {
  @override
  Widget build(BuildContext context) {
    return ViewModelBuilder<HomeViewModel>.reactive(
        viewModelBuilder: () => locator<HomeViewModel>(),
        onModelReady: (model) => model.init(),
        builder: (context, model, widget) {
          return WillPopScope(
            onWillPop: () async => !await model
                .navigatorKeys[model.currentTab].currentState
                .maybePop(),
            child: Scaffold(
              backgroundColor: Colors.white,
              body: Stack(
                children: <Widget>[
                  Offstage(
                    offstage: model.currentTab.index != 0,
                    child: CatalogTabNavigator(
                      navigatorKey: model.navigatorKeys[TabItem.catalog],
                    ),
                  ),
                  Offstage(
                    offstage: model.currentTab.index != 1,
                    child: SeedsTabNavigator(
```

```
                navigatorKey: model.navigatorKeys[TabItem.seeds],
              ),
            ),
            // Offstage(
            //   offstage: model.currentTab.index != 2,
            //   child: CalculatorTabNavigator(
            //     navigatorKey: model.navigatorKeys[TabItem.calculator],
            //   ),
            // ),
            Offstage(
              offstage: model.currentTab.index != 2,
              child: NewsTabNavigator(
                navigatorKey: model.navigatorKeys[TabItem.news],
              ),
            ),
            Offstage(
              offstage: model.currentTab.index != 3,
              child: ContactsTabNavigator(
                navigatorKey: model.navigatorKeys[TabItem.contacts],
              ),
            ),
            Align(
              alignment: Alignment.bottomCenter,
              child: BottomNavigation(
                currentTab: model.currentTab.index,
                onSelectTab: model.selectTab,
              ),
            ),
          ],
        ),
      ),
    );
  });
 }
}

import 'package:agrohelper/app/router.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:injectable/injectable.dart';
import 'package:stacked/stacked.dart';

enum TabItem { catalog, seeds, news, contacts }

@injectable
class HomeViewModel extends BaseViewModel {
 final HomeViewArguments arguments = Get.arguments;

 TabItem currentTab;

 Map<TabItem, GlobalKey<NavigatorState>> navigatorKeys = {
  TabItem.catalog: GlobalKey<NavigatorState>(),
  TabItem.seeds: GlobalKey<NavigatorState>(),
  TabItem.news: GlobalKey<NavigatorState>(),
  TabItem.contacts: GlobalKey<NavigatorState>()
 };

 init() {
```

```
    currentTab = arguments.tabItem;
  }

  void selectTab(int index) {
    currentTab = TabItem.values[index];
    notifyListeners();
  }
}
```

Додаток А4. Лістинг коду сторінки каталогу

```
import 'package:agrohelper/app/locator.dart';
import 'package:agrohelper/core/services/navigation/navigation_service.dart';
import 'package:agrohelper/core/utils/constants.dart';
import 'package:agrohelper/ui/shared/my_colors.dart';
import 'package:agrohelper/ui/views/catalog_details_view/catalog_details_view.dart';
import 'package:agrohelper/ui/views/search_view/search_view.dart';
import 'package:agrohelper/ui/widgets/catalog_item.dart';
import 'package:agrohelper/ui/widgets/my_app_bar.dart';
import 'package:easy_localization/easy_localization.dart';
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';

class CatalogScreen extends StatefulWidget {
  @override
  _CatalogScreenState createState() => _CatalogScreenState();
}

class _CatalogScreenState extends State<CatalogScreen> {
  final NavigationService _navigationService = locator<NavigationService>();

  void openNextScreen(int catalogId, String title, {bool cultures = false}) {
    Navigator.push(
        context,
        MaterialPageRoute(
            builder: (context) => (CatalogDetailsView(
                catalogId: catalogId,
                title: title,
                cultures: cultures,
              ))));
  }

  @override
  Widget build(BuildContext context) {
    return Stack(
      fit: StackFit.expand,
      children: <Widget>[
        Image.asset(
          "assets/background.png",
          fit: BoxFit.cover,
        ),
        Column(children: <Widget>[
          MyAppBar(
            title: tr("catalog"),
            expandedHeight: 174,
            onBack: () => _navigationService.pop(),
            optionIconSize: 45,
```

```
  optionIconColor: AppColors.darkBlue,
  onOption: () {
    Navigator.push(context,
        MaterialPageRoute(builder: (context) => SearchView()));
  },
  optionIconPath: "assets/Search.png",
),
Expanded(
  child: Padding(
    padding: const EdgeInsets.all(16),
    child: Column(
      children: <Widget>[
        Expanded(
          child: Row(
            mainAxisSize: MainAxisSize.min,
            children: <Widget>[
              Expanded(
                child: CatalogItem(
                  onPressed: () => openNextScreen(
                      Constants.gerbicidy, tr("herbicides")),
                  title: tr("herbicides"),
                  iconPath: "assets/Herbicides.png",
                  colors: [Color(0xFF2AAA28), Color(0xFFB5F1B4)],
                ),
              ),
              SizedBox(
                width: 10,
              ),
              Expanded(
                child: CatalogItem(
                  onPressed: () => openNextScreen(
                      Constants.protraviteli, tr("disinfectants")),
                  title: tr("disinfectants"),
                  iconPath: "assets/Disinfectants.png",
                  colors: [Color(0xFF4C97D8), Color(0xFF56CAEF)],
                ),
              ),
            ],
          ),
        ),
        Expanded(
          child: Row(
            mainAxisSize: MainAxisSize.min,
            children: <Widget>[
              Expanded(
                child: CatalogItem(
                  onPressed: () => openNextScreen(
                      Constants.insekticidy, tr("insecticides")),
                  title: tr("insecticides"),
                  iconPath: "assets/Insecticides.png",
                  colors: [
                    Color(0xFF0FC3CE),
                    Color.fromRGBO(187, 251, 255, 0.88)
                  ],
                ),
              ),
              SizedBox(
                width: 10,
```

```
      ),
      Expanded(
       child: CatalogItem(
        onPressed: () => openNextScreen(
          Constants.desikanty, tr("desiccants")),
        title: tr("desiccants"),
        iconPath: "assets/desiccants.png",
        colors: [Color(0xFFED8000), Color(0xFFFFAB6F)],
       ),
      ),
    ],
   ),
),
Expanded(
  child: Row(
    mainAxisSize: MainAxisSize.min,
    children: <Widget>[
      Expanded(
       child: CatalogItem(
        onPressed: () => openNextScreen(
          Constants.fungicidy, tr("fungicides")),
        title: tr("fungicides"),
        iconPath: "assets/Fungicides.png",
        colors: [Color(0xFF7E30A4), Color(0xFFDF9BFF)],
       ),
      ),
      SizedBox(
       width: 10,
      ),
      Expanded(
       child: CatalogItem(
        onPressed: () => openNextScreen(
          Constants.fumiganty, tr("fumigants")),
        title: tr("fumigants"),
        iconPath: "assets/Fumigant.png",
        colors: [Color(0xFF5E5E5E), Color(0xFFA6A6A6)],
       ),
      ),
    ],
  ),
),
Expanded(
  child: Row(
    mainAxisSize: MainAxisSize.min,
    children: <Widget>[
      Expanded(
       child: CatalogItem(
        onPressed: () => openNextScreen(
          Constants.smachivatel, tr("wetting_agents")),
        title: tr("wetting_agents"),
        iconPath: "assets/Adjuvants.png",
        colors: [Color(0xFFC93B3B), Color(0xFFFF9494)],
       ),
      ),
      SizedBox(
       width: 10,
      ),
      Expanded(
```

```
                    child: CatalogItem(
                      onPressed: () => openNextScreen(
                          Constants.gerbicidy, tr("microfertilizers"),
                          cultures: true),
                      title: tr("microfertilizers"),
                      iconPath: "assets/Microfertilizers.png",
                      colors: [Color(0xFFE9C713), Color(0xFFEAE109)],
                    ),
                  ),
                ],
              ),
            )
          ],
        ),
      ),
      ),
      SizedBox(
        height: 80,
      )
    ]),
  ],
 );
 }
}
```

Додаток А5. Лістинг коду списку препаратів

```
import 'package:agrohelper/app/locator.dart';
import 'package:agrohelper/core/model/culture_model.dart';
import 'package:agrohelper/core/model/default_model.dart';
import 'package:agrohelper/ui/widgets/items/catalog_item.dart';
import 'package:agrohelper/ui/widgets/my_sliver_app_bar.dart';
import 'package:agrohelper/ui/widgets/my_sliver_app_header.dart';
import 'package:flutter/material.dart';
import 'package:stacked/stacked.dart';

import 'catalog_details_view_model.dart';

class CatalogDetailsView extends StatefulWidget {
 final int catalogId;
 final String title;
 final bool cultures;

 const CatalogDetailsView(
    {Key key, this.catalogId, @required this.title, this.cultures = false})
    : super(key: key);

 @override
 _CatalogDetailsViewState createState() => _CatalogDetailsViewState();
}

class _CatalogDetailsViewState extends State<CatalogDetailsView> {
 @override
 Widget build(BuildContext context) {
  return ViewModelBuilder<CatalogDetailsViewModel>.reactive(
      viewModelBuilder: () => locator<CatalogDetailsViewModel>(),
      onModelReady: (model) =>
```

```
          model.init(widget.cultures, widget.catalogId, context),
        builder: (context, model, widget) {
         return Stack(
          fit: StackFit.expand,
          children: <Widget>[
           Image.asset(
            "assets/background.png",
            fit: BoxFit.cover,
           ),
           NestedScrollView(
            headerSliverBuilder:
              (BuildContext context, bool innerBoxIsScrolled) => <Widget>[
             SliverPersistentHeader(
              pinned: false, delegate: MySliverAppHeader()),
             SliverPersistentHeader(
              pinned: true,
              delegate: MySliverAppBar(
               onBack: model.onBack,
               optionIconSize: 45,
               onOption: model.onOption,
               optionIconPath: "assets/Filter.png",
               title: this.widget.title,
              ))
            ],
            body: model.cultures ? CatalogBody() : DefaultBody(),
           ),
          ],
         );
        });
  }
}

class CatalogBody extends ViewModelWidget<CatalogDetailsViewModel> {
 @override
 Widget build(BuildContext context, CatalogDetailsViewModel viewModel) {
  return StreamBuilder<List<CultureModel>>(
     stream: viewModel.culture,
     builder: (context, snapshot) {
      List<CultureModel> list = snapshot.data ?? [];
      return ListView.builder(
       itemCount: list.length,
       padding: EdgeInsets.only(right: 15, left: 15, bottom: 80, top: 10),
       physics: BouncingScrollPhysics(),
       itemBuilder: (context, index) {
        CultureModel item = list[index];
        return CatalogItem(
         title: item.title,
         description: item.decriptionCustom,
         onPressed: () =>
           viewModel.openCatalogItemScreen<CultureModel>(item),
        );
       },
      );
     });
  }
}

class DefaultBody extends ViewModelWidget<CatalogDetailsViewModel> {
```

```
  @override
  Widget build(BuildContext context, CatalogDetailsViewModel viewModel) {
    return StreamBuilder<List<DefaultModel>>(
        stream: viewModel.catalog,
        builder: (context, snapshot) {
          List<DefaultModel> list = snapshot.data ?? [];
          return ListView.builder(
            itemCount: list.length,
            padding: EdgeInsets.only(right: 15, left: 15, bottom: 80, top: 10),
            physics: BouncingScrollPhysics(),
            itemBuilder: (context, index) {
              DefaultModel item = list[index];
              return CatalogItem(
                title: item.title,
                description: item.ko,
                onPressed: () =>
                    viewModel.openCatalogItemScreen<DefaultModel>(item),
              );
            },
          );
        });
  }
}
import 'package:agrohelper/core/model/culture_model.dart';
import 'package:agrohelper/core/model/default_model.dart';
import 'package:agrohelper/core/services/database/database_service.dart';
import 'package:agrohelper/core/singleton/filter_singleton.dart';
import 'package:agrohelper/ui/views/catalog/catalog_item_screen.dart';
import 'package:agrohelper/ui/views/filter_view/filter_view.dart';
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:injectable/injectable.dart';
import 'package:rxdart/rxdart.dart';
import 'package:stacked/stacked.dart';

@injectable
class CatalogDetailsViewModel extends BaseViewModel {
  final DatabaseService _databaseService;
  final FilterSingleton _filterSingleton;

  final _catalogFetcher = PublishSubject<List<DefaultModel>>();
  final _culturesFetcher = PublishSubject<List<CultureModel>>();

  CatalogDetailsViewModel(this._databaseService, this._filterSingleton);

  Stream<List<DefaultModel>> get catalog => _catalogFetcher.stream;

  Stream<List<CultureModel>> get culture => _culturesFetcher.stream;

  bool cultures;
  int catalogId;
  BuildContext context;

  init(bool cultures, int catalogId, BuildContext context) {
    this.cultures = cultures;
    this.catalogId = catalogId;
    this.context = context;
    fetchData();
```

```
  }

  void onOption() {
    Navigator.push(
        context,
        MaterialPageRoute(
            builder: (context) => FilterScreen(
                isMicro: cultures,
              ))).then((value) {
      fetchData();
    });
  }

  void onBack() {
    _filterSingleton.reset();
    Navigator.of(context).pop();
  }

  void openCatalogItemScreen<T>(T model) {
    Navigator.push(
        context,
        MaterialPageRoute(
            builder: (context) => CatalogItemScreen<T>(
                model: model,
              )));
  }

  void _fetchCatalog(int catalogId, {int szrTopics, int szrActions}) async {
    List<DefaultModel> contactsList =
        await _databaseService.readCatalog(catalogId, szrTopics, szrActions);
    _catalogFetcher.sink.add(contactsList);
  }

  void _fetchCultures(int microTopics) async {
    List<CultureModel> contactsList =
        await _databaseService.readCultures(microTopics);
    _culturesFetcher.sink.add(contactsList);
  }

  void fetchData() {
    if (cultures) {
      int microTopics = _filterSingleton.microTopics;
      _fetchCultures(microTopics);
    } else {
      int szrTopics = _filterSingleton.szrTopics;
      int szrActions = _filterSingleton.szrActions;
      _fetchCatalog(catalogId, szrTopics: szrTopics, szrActions: szrActions);
    }
  }

  @override
  void dispose() {
    _culturesFetcher.close();
    _catalogFetcher.close();
    super.dispose();
  }
}
```

Додаток А6. Лістинг коду екрану препарату

```
import 'dart:convert';

import 'package:agrohelper/ui/shared/my_colors.dart';
import 'package:agrohelper/ui/widgets/my_sliver_app_bar.dart';
import 'package:agrohelper/ui/widgets/my_sliver_app_header.dart';
import 'package:flutter/foundation.dart';
import 'package:flutter/gestures.dart';
import 'package:flutter/material.dart';
import 'package:flutter_html/flutter_html.dart';
import 'package:flutter_html/html_parser.dart';
import 'package:flutter_html/style.dart';
import 'package:webview_flutter/webview_flutter.dart';
import 'package:webview_flutter_plus/webview_flutter_plus.dart';

class CatalogItemScreen<T> extends StatefulWidget {
  final T model;

  const CatalogItemScreen({Key key, @required this.model}) : super(key: key);

  @override
  _CatalogItemScreenState createState() => _CatalogItemScreenState();
}

class _CatalogItemScreenState extends State<CatalogItemScreen> {
  bool isExpanded = false;
  double height = 100;
  Map<String, double> tablesMap = Map<String, double>();

  final Color contentColor = Color.fromRGBO(187, 251, 255, 0.88);
  final Color background = Color.fromRGBO(92, 132, 138, 0.75);

  @override
  void dispose() {
    super.dispose();
  }

  Widget drawTable(
    RenderContext context,
    Widget parsedChild,
  ) {
    String style = "";
    bool isWidth = false;
    bool isClass = false;
    var innerHtml = context.tree.element.innerHtml;
    var attributes = context.tree.attributes;
    attributes.forEach((key, value) {
      String field = key.toString() + "=\"" + value + "\" ";
      if (key == "style") {
        field = 'style="width: 100%;"';
        isWidth = true;
      }
      if (key == "width") {
        field = '';
      }
```

```
  if (key == "class") {
    field = '';
    isClass = true;
  }
  style += field;
 });
 if (!isWidth) {
  String field = 'style="width: 100%;"';
  style += field;
 }
 if (!isClass) {
  String field = '';
  style += field;
 }
 WebViewPlusController _controller;
 return Container(
  height: tablesMap[innerHtml] ?? 150,
  width: MediaQuery.of(this.context).size.width,
  child: Offstage(
   offstage: tablesMap[innerHtml] == null,
   child: ClipRRect(
    borderRadius: BorderRadius.all(Radius.circular(10)),
    child: WebViewPlus(
     initialUrl: Uri.dataFromString(
          '<table id="table" $style" height="100%">' +
            innerHtml +
            "</table>",
          mimeType: 'text/html',
          encoding: Encoding.getByName('utf-8'))
        .toString(),
     onWebViewCreated: (webViewController) {
      _controller = webViewController;
     },
     javascriptMode: JavascriptMode.unrestricted,
     gestureNavigationEnabled: true,
     gestureRecognizers: <Factory<OneSequenceGestureRecognizer>>[
      Factory<OneSequenceGestureRecognizer>(
       () => EagerGestureRecognizer(),
      ),
     ].toSet(),
     onPageFinished: (url) async {
      double height = await _controller.getHeight();
      if (!tablesMap.keys.contains(innerHtml)) {
       setState(() {
        tablesMap[innerHtml] = height;
       });
      }
     },
    ),
   ),
  ),
 );
}

@override
Widget build(BuildContext context) {
 return Scaffold(
  body: Stack(
```

```
      fit: StackFit.expand,
      children: <Widget>[
       Image.asset(
         "assets/background.png",
         fit: BoxFit.cover,
       ),
       NestedScrollView(
         headerSliverBuilder:
            (BuildContext context, bool innerBoxIsScrolled) => <Widget>[
          SliverPersistentHeader(
             pinned: false, delegate: MySliverAppHeader()),
          SliverPersistentHeader(
             pinned: true,
             delegate: MySliverAppBar(
               title: widget.model.title,
             ))
         ],
         body: ListView(
          padding:
             EdgeInsets.only(bottom: 80, top: 10, right: 10, left: 10),
          physics: BouncingScrollPhysics(),
          children: <Widget>[
           Html(
             data: widget.model.content,
             customRender: {"table": drawTable},
             style: {
               "p": Style(
                  fontSize: FontSize(13),
                  color: AppColors.lightBlack,
                  textAlign: TextAlign.justify,
                  fontFamily: 'OpenSansRegular'),
               "strong": Style(
                  fontSize: FontSize(14),
                  color: AppColors.black,
                  fontFamily: 'OpenSansBold'),
               "div": Style(
                  fontSize: FontSize(13),
                  color: AppColors.lightBlack,
                  textAlign: TextAlign.justify,
                  fontFamily: 'OpenSansRegular'),
               "li": Style(
                  fontSize: FontSize(13),
                  margin: EdgeInsets.only(bottom: 5),
                  color: AppColors.lightBlack,
                  textAlign: TextAlign.justify,
                  fontFamily: 'OpenSansRegular'),
             },
           )
          ],
         ),
        ),
       ],
      ),
     );
   }
}
```

Додаток А7. Лістинг коду екрану фільтрації

```
import 'package:agrohelper/app/locator.dart';
import 'package:agrohelper/ui/shared/my_colors.dart';
import 'package:agrohelper/ui/views/filter_view/filter_view_model.dart';
import 'package:agrohelper/ui/widgets/contacts_item.dart';
import 'package:agrohelper/ui/widgets/my_app_bar.dart';
import 'package:agrohelper/ui/widgets/sq_picker.dart';
import 'package:easy_localization/easy_localization.dart';
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:stacked/stacked.dart';

class FilterScreen extends StatefulWidget {
  final bool isMicro;

  const FilterScreen({Key key, this.isMicro = false}) : super(key: key);

  @override
  _FilterScreenState createState() => _FilterScreenState();
}

class _FilterScreenState extends State<FilterScreen> {
  @override
  Widget build(BuildContext context) {
    return ViewModelBuilder<FilterViewModel>.reactive(
        viewModelBuilder: () => locator<FilterViewModel>(),
        onModelReady: (model) => model.init(widget.isMicro),
        builder: (context, model, widget) {
          return Scaffold(
            body: Stack(
              fit: StackFit.expand,
              children: <Widget>[
                Image.asset(
                  "assets/background.png",
                  fit: BoxFit.cover,
                ),
                Column(
                  children: <Widget>[
                    MyAppBar(
                      title: tr("filter"),
                      expandedHeight: 174,
                    ),
                    SizedBox(
                      height: 8,
                    ),
                    ContactsItem(
                      iconPath: "assets/arrow_down.png",
                      onPressed: () {
                        SqPicker(context,
                            backgroundColor: AppColors.blue,
                            buttonTextStyle: TextStyle(
                                color: Colors.white,
                                fontSize: 15,
                                fontFamily: 'OpenSansRegular'),
                            itemList: model.firstList
                                .map((filter) =>
                                    Center(child: Text(filter.name)))
                                .toList(),
```

```
            index: model.index1,
            onPickerChanged: model.onFirstPickerChanged)
          .show();
      },
      title: model.text1,
    ),
    SizedBox(
      height: 8,
    ),
    Visibility(
      visible: !model.isMicro,
      child: ContactsItem(
        iconPath: "assets/arrow_down.png",
        onPressed: () {
          SqPicker(context,
              backgroundColor: AppColors.blue,
              buttonTextStyle: TextStyle(
                  color: Colors.white,
                  fontSize: 15,
                  fontFamily: 'OpenSansRegular'),
              itemList: model.secondList
                  .map((filter) =>
                      Center(child: Text(filter.name)))
                  .toList(),
              index: model.index2,
              onPickerChanged: model.onSecondPickerChanged)
            .show();
        },
        title: model.text2,
      ),
    ),
    Expanded(
      child: Container(),
    ),
    Padding(
      padding: const EdgeInsets.only(bottom: 75),
      child: TextButton(
          style: TextButton.styleFrom(
            backgroundColor: AppColors.blue,
            padding: EdgeInsets.all(0),
            shape: RoundedRectangleBorder(
                borderRadius:
                    BorderRadius.all(Radius.circular(55))),
          ),
          onPressed: () => Navigator.of(context).pop(),
          child: Container(
              width: 300,
              height: 50,
              child: Center(
                  child: Text(tr("apply"),
                      style: TextStyle(
                          color: Colors.white,
                          fontSize: 16,
                          fontFamily: 'OpenSansSemiBold'))))),
    )
  ],
 ),
],
```

```
          ),
        );
      });
  }
}

import 'package:agrohelper/core/model/topic_model.dart';
import 'package:agrohelper/core/services/database/database_service.dart';
import 'package:agrohelper/core/singleton/filter_singleton.dart';
import 'package:agrohelper/core/utils/constants.dart';
import 'package:easy_localization/easy_localization.dart';
import 'package:injectable/injectable.dart';
import 'package:rxdart/rxdart.dart';
import 'package:stacked/stacked.dart';

@injectable
class FilterViewModel extends BaseViewModel {
  final DatabaseService _databaseService;
  final FilterSingleton _filterSingleton;

  final _filterFetcher = PublishSubject<FilterModel>();
  final _microTopicsFetcher = PublishSubject<List<TopicModel>>();

  FilterViewModel(this._databaseService, this._filterSingleton);

  Stream<FilterModel> get filter => _filterFetcher.stream;

  Stream<List<TopicModel>> get microTopics => _microTopicsFetcher.stream;

  String _baseText1 = tr("cultivated_crops");
  String _baseText2 = tr("active_substances");

  String text1;
  String text2;

  int index1 = 0;
  int index2 = 0;

  List<TopicModel> firstList = [];
  List<TopicModel> secondList = [];

  bool isMicro;

  void init(bool isMicro) {
    this.isMicro = isMicro;
    if (!isMicro) {
      text1 = _baseText1;
      text2 = _baseText2;
      index1 = _filterSingleton.szrTopicsIndex;
      index2 = _filterSingleton.szrActionsIndex;
      fetchFilter();
      filter.listen((filter) {
        firstList.addAll(filter.szrTopics);
        secondList.addAll(filter.szrActions);
        firstList.insert(0, TopicModel(name: tr("select")));
        secondList.insert(0, TopicModel(name: tr("select")));
        if (index1 > 0) {
          text1 = firstList[index1].name;
```

```
      }
      if (index2 > 0) {
        text2 = secondList[index2].name;
      }
      notifyListeners();
    });
  } else {
    _baseText1 = tr("culture");
    text1 = _baseText1;
    index1 = FilterSingleton().microTopicsIndex;
    fetchMicroTopics();
    microTopics.listen((filter) {
      firstList.addAll(filter);
      firstList.insert(0, TopicModel(name: tr("select")));
      if (index1 > 0) {
        text1 = firstList[index1].name;
      }
      notifyListeners();
    });
  }
}

void onFirstPickerChanged(int index) {
  index1 = index;
  if (!isMicro) {
    FilterSingleton().szrTopicsIndex = index1;
  } else {
    FilterSingleton().microTopicsIndex = index1;
  }
  if (index1 > 0) {
    text1 = firstList[index1].name;
    if (isMicro) {
      FilterSingleton().microTopics = firstList[index1].id;
    } else {
      FilterSingleton().szrTopics = firstList[index1].id;
    }
  } else {
    FilterSingleton().szrTopics = null;
    FilterSingleton().microTopics = null;
    text1 = _baseText1;
  }
  notifyListeners();
}

void onSecondPickerChanged(int index) {
  index2 = index;
  FilterSingleton().szrActionsIndex = index2;
  if (index2 > 0) {
    text2 = secondList[index2].name;
    FilterSingleton().szrActions = secondList[index2].id;
  } else {
    text2 = _baseText2;
    FilterSingleton().szrActions = null;
  }
  notifyListeners();
}

fetchFilter() async {
```

```
    TopicBox szrActions = await _databaseService.readDefault<TopicBox>(
        Constants.szrActions, Constants.topics);
    TopicBox szrTopics = await _databaseService.readDefault<TopicBox>(
        Constants.szrTopics, Constants.topics);
    _filterFetcher.sink.add(
        FilterModel(szrActions: szrActions.list, szrTopics: szrTopics.list));
  }

  fetchMicroTopics() async {
    TopicBox microTopics = await _databaseService.readDefault<TopicBox>(
        Constants.microTopics, Constants.topics);
    _microTopicsFetcher.sink.add(microTopics.list);
  }

  @override
  void dispose() {
    _microTopicsFetcher.close();
    _filterFetcher.close();
    super.dispose();
  }
}

class FilterModel {
  List<TopicModel> szrTopics;
  List<TopicModel> szrActions;

  FilterModel({this.szrTopics, this.szrActions});
}
```

Додаток А8. Лістинг коду екрану контактів

```
import 'package:agrohelper/app/locator.dart';
import 'package:agrohelper/ui/views/contacts/contacts_bloc.dart';
import 'package:agrohelper/core/model/contacts_model.dart';
import 'package:agrohelper/core/services/navigation/navigation_service.dart';
import 'package:agrohelper/ui/views/contacts/contacts_details_screen.dart';
import 'package:agrohelper/ui/widgets/contacts_item.dart';
import 'package:agrohelper/ui/widgets/my_sliver_app_bar.dart';
import 'package:agrohelper/ui/widgets/my_sliver_app_header.dart';
import 'package:easy_localization/easy_localization.dart';
import 'package:flutter/material.dart';

class ContactsScreen extends StatefulWidget {
  const ContactsScreen({Key key}) : super(key: key);

  @override
  _ContactsScreenState createState() => _ContactsScreenState();
}

class _ContactsScreenState extends State<ContactsScreen> {
  final ContactsBloc contactsBloc = ContactsBloc();
  final NavigationService _navigationService = locator<NavigationService>();

  @override
  void initState() {
    super.initState();
    contactsBloc.fetchContacts();
```

```
}

void onPush(ContactModel contactModel) {
  Navigator.push(
      context,
      MaterialPageRoute(
          builder: (context) => ContactsDetailsScreens(
              contactsModel: contactModel,
            )));
}

@override
void dispose() {
  contactsBloc.dispose();
  super.dispose();
}

@override
Widget build(BuildContext context) {
  return Stack(
    fit: StackFit.expand,
    children: <Widget>[
      Image.asset(
        "assets/background.png",
        fit: BoxFit.cover,
      ),
      NestedScrollView(
        headerSliverBuilder:
            (BuildContext context, bool innerBoxIsScrolled) => <Widget>[
          SliverPersistentHeader(
              pinned: false, delegate: MySliverAppHeader()),
          SliverPersistentHeader(
              pinned: true,
              delegate: MySliverAppBar(
                onBack: () => _navigationService.pop(),
                title: tr("contacts"),
              ))
        ],
        body: StreamBuilder<List<ContactModel>>(
            stream: contactsBloc.contacts,
            builder: (context, snapshot) {
              List<ContactModel> contacts = snapshot.data ?? [];
              return ListView.builder(
                itemCount: contacts.length,
                padding: EdgeInsets.only(top: 17, bottom: 80),
                physics: BouncingScrollPhysics(),
                itemBuilder: (context, index) {
                  ContactModel contactModel = contacts[index];
                  return ContactsItem(
                    onPressed: () => onPush(contactModel),
                    title: contactModel.city,
                  );
                },
              );
            }),
      ),
    ],
  );
```

```
  }
}

import 'package:agrohelper/core/model/contacts_model.dart';
import 'package:agrohelper/ui/widgets/items/address_item.dart';
import 'package:agrohelper/ui/widgets/items/contact_action_item.dart';
import 'package:agrohelper/ui/widgets/map_widget.dart';
import 'package:agrohelper/ui/widgets/my_sliver_app_bar.dart';
import 'package:agrohelper/ui/widgets/my_sliver_app_header.dart';
import 'package:easy_localization/easy_localization.dart';
import 'package:flutter/cupertino.dart';
import 'package:flutter/foundation.dart';
import 'package:flutter/material.dart';
import 'package:url_launcher/url_launcher.dart';

class ContactsDetailsScreens extends StatefulWidget {
  final ContactModel contactsModel;

  const ContactsDetailsScreens({Key key, @required this.contactsModel})
      : super(key: key);

  @override
  _ContactsDetailsScreensState createState() => _ContactsDetailsScreensState();
}

class _ContactsDetailsScreensState extends State<ContactsDetailsScreens> {
  @override
  Widget build(BuildContext context) {
    return Stack(
      fit: StackFit.expand,
      children: <Widget>[
        Image.asset(
          "assets/background.png",
          fit: BoxFit.cover,
        ),
        NestedScrollView(
          headerSliverBuilder:
              (BuildContext context, bool innerBoxIsScrolled) => <Widget>[
            SliverPersistentHeader(
                pinned: false, delegate: MySliverAppHeader()),
            SliverPersistentHeader(
                pinned: true,
                delegate: MySliverAppBar(
                  title: widget.contactsModel.city,
                ))
          ],
          body: ListView.builder(
            physics: BouncingScrollPhysics(),
            padding: EdgeInsets.only(top: 16, bottom: 80),
            itemCount: widget.contactsModel.addresses.length,
            itemBuilder: (BuildContext context, int index) {
              Address address = widget.contactsModel.addresses[index];
              return buildAddressWidget(address);
            },
          ),
        ),
      ],
    );
```

```
}

_launchURL(String url) async {
  if (await canLaunch(url)) {
    await launch(url);
  }
}

Widget buildAddressWidget(Address address) {
  List<Widget> list = [];
  if (address.address != null) {
    list.add(Padding(
      padding: const EdgeInsets.only(bottom: 1),
      child: AddressItem(
        address: tr("address"),
        content: address.address,
      ),
    ));
  }
  if (address.phone != null) {
    address.phone.forEach((element) {
      list.add(Padding(
        padding: const EdgeInsets.only(bottom: 1),
        child: ContactActionItem(
          onTab: () {
            _launchURL("tel:$element");
          },
          title: tr("phone"),
          buttonText: tr("call"),
          content: element,
          pathIcon: "assets/Phone.png",
        ),
      ));
    });
  }
  if (address.email != null) {
    list.add(Padding(
      padding: const EdgeInsets.only(bottom: 1),
      child: ContactActionItem(
        onTab: () {
          _launchURL("mailto:${address.email}");
        },
        title: "E-mail",
        buttonText: tr("write"),
        content: address.email,
        pathIcon: "assets/Email.png",
      ),
    ));
  }
  if (address.address != null &&
      address.latitude != null &&
      address.longitude != null) {
    list.add(Padding(
      padding: const EdgeInsets.only(bottom: 1),
      child: MapWidget(
        latitude: address.latitude,
        longitude: address.longitude,
        markerId: address.address,
```

```
      ),
    ));
   }
   return Column(children: list);
  }
}

import 'package:agrohelper/app/locator.dart';
import 'package:agrohelper/core/bloc.dart';
import 'package:agrohelper/core/model/contacts_model.dart';
import 'package:agrohelper/core/services/database/database_service.dart';
import 'package:rxdart/rxdart.dart';

class ContactsBloc extends Bloc {
  final DatabaseService _databaseService = locator<DatabaseService>();
  final _contactsFetcher = PublishSubject<List<ContactModel>>();

  Stream<List<ContactModel>> get contacts => _contactsFetcher.stream;

  fetchContacts() async {
    List<ContactModel> contactsList = await _databaseService.fetchContacts();
    _contactsFetcher.sink.add(contactsList);
  }

  @override
  void dispose() {
    _contactsFetcher.close();
  }
}

import 'package:agrohelper/ui/shared/my_colors.dart';
import 'package:flutter/material.dart';

class AddressItem extends StatelessWidget {
  final String address;
  final String content;

  final Color addressColor;
  final Color contentColor;
  final Color background;
  final bool isBold;

  const AddressItem({
    Key key,
    @required this.content,
    this.address,
    this.addressColor = AppColors.black,
    this.contentColor = AppColors.lightBlack,
    this.background = AppColors.background,
    this.isBold = false,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Container(
      width: MediaQuery.of(context).size.width,
      color: background,
      padding: EdgeInsets.symmetric(vertical: 8, horizontal: 16),
```

```
    child: Column(
     mainAxisAlignment: MainAxisAlignment.center,
     crossAxisAlignment: CrossAxisAlignment.start,
     children: <Widget>[
      Text(
       address,
       style: TextStyle(
          fontSize: 16,
          fontWeight: isBold ? FontWeight.bold : null,
          fontFamily: 'OpenSansRegular',
          color: addressColor),
      ),
      SizedBox(
       height: 5,
      ),
      Text(
       content,
       style: TextStyle(
          fontSize: 13,
          fontFamily: 'OpenSansRegular',
          color: contentColor),
      ),
     ],
    ),
   );
  }
}

import 'package:agrohelper/ui/shared/my_colors.dart';
import 'package:flutter/material.dart';

class ContactActionItem extends StatelessWidget {
 final String title;
 final String content;
 final String pathIcon;
 final String buttonText;
 final VoidCallback onTab;

 const ContactActionItem({
  Key key,
  this.title,
  this.content,
  this.pathIcon,
  this.buttonText,
  this.onTab,
 }) : super(key: key);

 @override
 Widget build(BuildContext context) {
  return Container(
   height: 80,
   color: AppColors.background,
   padding: EdgeInsets.symmetric(vertical: 8, horizontal: 16),
   child: Row(
    children: <Widget>[
     Expanded(
      child: Column(
       mainAxisAlignment: MainAxisAlignment.center,
```

```
            crossAxisAlignment: CrossAxisAlignment.start,
            children: <Widget>[
              Text(
                title,
                style: TextStyle(
                    fontSize: 16,
                    fontFamily: 'OpenSansRegular',
                    color: AppColors.black),
              ),
              SizedBox(
                height: 5,
              ),
              Text(
                content,
                style: TextStyle(
                    fontSize: 13,
                    fontFamily: 'OpenSansRegular',
                    color: AppColors.lightBlack),
              ),
            ],
          ),
        ),
        TextButton(
            onPressed: onTab,
            style: TextButton.styleFrom(
              backgroundColor: AppColors.blue,
              shape: RoundedRectangleBorder(
                  borderRadius: BorderRadius.all(Radius.circular(20))),
            ),
            child: Container(
              height: 25,
              width: 125,
              child: Row(
                mainAxisAlignment: MainAxisAlignment.spaceBetween,
                crossAxisAlignment: CrossAxisAlignment.center,
                children: <Widget>[
                  Image.asset(
                    pathIcon,
                    height: 20,
                    width: 20,
                  ),
                  Expanded(
                      child: Text(
                    buttonText,
                    textAlign: TextAlign.center,
                    style: TextStyle(
                        color: Colors.white,
                        fontSize: 12,
                        fontFamily: 'OpenSansSemiBold'),
                  ))
                ],
              ),
            ))
      ],
    ),
  );
 }
}
```

Додаток А9. Лістинг коду екрану новин

```
import 'package:agrohelper/app/locator.dart';
import 'package:agrohelper/ui/views/news/news_bloc.dart';
import 'package:agrohelper/core/model/news_model.dart';
import 'package:agrohelper/core/services/navigation/navigation_service.dart';
import 'package:agrohelper/ui/views/news/news_details_screen.dart';
import 'package:agrohelper/ui/widgets/items/news_item.dart';
import 'package:agrohelper/ui/widgets/my_sliver_app_bar.dart';
import 'package:agrohelper/ui/widgets/my_sliver_app_header.dart';
import 'package:easy_localization/easy_localization.dart';
import 'package:flutter/material.dart';

class NewsScreen extends StatefulWidget {
  const NewsScreen({Key key}) : super(key: key);

  @override
  _NewsScreenState createState() => _NewsScreenState();
}

class _NewsScreenState extends State<NewsScreen> {
  final NewsBloc newsBloc = NewsBloc();
  final NavigationService _navigationService = locator<NavigationService>();
  final GlobalKey<RefreshIndicatorState> _refreshIndicatorKey =
      new GlobalKey<RefreshIndicatorState>();

  List<NewsModel> news = [];

  @override
  void initState() {
    super.initState();
    WidgetsBinding.instance
        .addPostFrameCallback((_) => _refreshIndicatorKey.currentState.show());
    newsBloc.news.listen((news) {
      setState(() {
        this.news = news;
      });
    });
  }

  Future<dynamic> _refresh() {
    return newsBloc.fetchNews();
  }

  @override
  void dispose() {
    newsBloc.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Stack(
      fit: StackFit.expand,
      children: <Widget>[
        Image.asset(
```

```
        "assets/background.png",
        fit: BoxFit.cover,
      ),
      NestedScrollView(
        headerSliverBuilder:
          (BuildContext context, bool innerBoxIsScrolled) => <Widget>[
              SliverPersistentHeader(
                pinned: false, delegate: MySliverAppHeader()),
              SliverPersistentHeader(
                pinned: true,
                delegate: MySliverAppBar(
                  title: tr("news"),
                  onBack: () => _navigationService.pop(),
                ))
            ],
        body: RefreshIndicator(
          key: _refreshIndicatorKey,
          onRefresh: _refresh,
          child: ListView.builder(
            itemCount: news.length,
            padding:
                EdgeInsets.only(right: 15, left: 15, top: 15, bottom: 80),
            physics: BouncingScrollPhysics(),
            itemBuilder: (context, index) {
              NewsModel newsModel = news[index];
              return NewsItem(
                onPressed: () {
                  Navigator.push(
                    context,
                    MaterialPageRoute(
                      builder: (context) => NewsDetailsScreens(
                          newsModel: newsModel,
                        )));
                },
                title: newsModel.title.rendered,
                description: newsModel.excerpt.rendered,
                publishDate: newsModel.date,
              );
            },
          ),
        )),
    ],
   );
  }
}

import 'package:agrohelper/core/model/news_model.dart';
import 'package:agrohelper/ui/shared/my_colors.dart';
import 'package:agrohelper/ui/widgets/my_sliver_app_bar.dart';
import 'package:agrohelper/ui/widgets/my_sliver_app_header.dart';
import 'package:flutter/material.dart';
import 'package:flutter_html/flutter_html.dart';
import 'package:flutter_html/style.dart';
import 'package:url_launcher/url_launcher.dart';

class NewsDetailsScreens extends StatefulWidget {
  final NewsModel newsModel;
```

```
  const NewsDetailsScreens({Key key, this.newsModel}) : super(key: key);

  @override
  _NewsDetailsScreensState createState() => _NewsDetailsScreensState();
}

class _NewsDetailsScreensState extends State<NewsDetailsScreens> {
  _launchURL(String url) async {
    if (await canLaunch(url)) {
      await launch(url);
    }
  }

  @override
  Widget build(BuildContext context) {
    return Stack(
      fit: StackFit.expand,
      children: <Widget>[
        Image.asset(
          "assets/background.png",
          fit: BoxFit.cover,
        ),
        NestedScrollView(
          headerSliverBuilder:
              (BuildContext context, bool innerBoxIsScrolled) => <Widget>[
            SliverPersistentHeader(
                pinned: false, delegate: MySliverAppHeader()),
            SliverPersistentHeader(
                pinned: true,
                delegate: MySliverAppBar(
                  title: widget.newsModel.title.rendered,
                ))
          ],
          body: Container(
            decoration: BoxDecoration(
                color: AppColors.background,
                borderRadius: BorderRadius.all(Radius.circular(10))),
            margin: EdgeInsets.only(right: 7, left: 7, bottom: 80, top: 10),
            child: ListView(
              physics: BouncingScrollPhysics(),
              padding:
                  EdgeInsets.only(top: 10, left: 10, right: 10, bottom: 10),
              children: <Widget>[
                Html(
                  data: widget.newsModel.content.rendered,
                  style: {
                    "p": Style(
                        fontSize: FontSize(13),
                        color: AppColors.lightBlack,
                        textAlign: TextAlign.justify,
                        fontFamily: 'OpenSansRegular'),
                    "strong": Style(
                        fontSize: FontSize(16),
                        color: AppColors.black,
                        fontFamily: 'OpenSansBold'),
                    "a": Style(
                        fontSize: FontSize(13),
                        color: AppColors.blue,
```

```
                fontFamily: 'OpenSansBold'),
              "img": Style(
                alignment: Alignment.center,
              ),
              "div": Style(
                fontSize: FontSize(13),
                color: AppColors.lightBlack,
                textAlign: TextAlign.justify,
                fontFamily: 'OpenSansRegular'),
              "li": Style(
                fontSize: FontSize(13),
                margin: EdgeInsets.only(bottom: 5),
                color: AppColors.lightBlack,
                textAlign: TextAlign.justify,
                fontFamily: 'OpenSansRegular'),
            },
            onLinkTap: (url, context, attributes, element) => _launchURL(url),
            // onLinkTap: _launchURL,
          )
        ],
      ),
    ),
  ),
 ],
 );
 }
}

import 'package:agrohelper/app/locator.dart';
import 'package:agrohelper/core/bloc.dart';
import 'package:agrohelper/core/model/news_model.dart';
import 'package:agrohelper/core/services/wordpress/wordpress_service.dart';
import 'package:rxdart/rxdart.dart';

class NewsBloc extends Bloc {
 final WordPressService _wordPressService = locator<WordPressService>();
 final _newsFetcher = PublishSubject<List<NewsModel>>();

 Stream<List<NewsModel>> get news => _newsFetcher.stream;

 fetchNews() async {
  List<NewsModel> newsList = await _wordPressService.fetchNews();
  _newsFetcher.sink.add(newsList);
 }

 @override
 void dispose() {
  _newsFetcher.close();
 }
}

import 'package:agrohelper/ui/shared/my_colors.dart';
import 'package:flutter/material.dart';
import 'package:flutter_html/flutter_html.dart';
import 'package:flutter_html/style.dart';
import 'package:html_unescape/html_unescape.dart';
import 'package:intl/date_symbol_data_local.dart';
import 'package:intl/intl.dart';
```

```
class NewsItem extends StatelessWidget {
  final String title;
  final String description;
  final DateTime publishDate;

  const NewsItem({
    Key key,
    @required this.onPressed,
    @required this.title,
    @required this.description,
    @required this.publishDate,
  }) : super(key: key);

  final VoidCallback onPressed;

  String formatDate(DateTime publishDate) {
    initializeDateFormatting("ru");
    DateFormat dateFormat = DateFormat('MMM dd, yyyy', "ru");
    return dateFormat.format(publishDate);
  }

  String formatTitle(String title) {
    var unescape = new HtmlUnescape();
    var text = unescape.convert(title);
    return text;
  }

  @override
  Widget build(BuildContext context) {
    return Padding(
      padding: EdgeInsets.only(bottom: 10),
      child: TextButton(
          onPressed: onPressed,
          style: TextButton.styleFrom(
            shape: RoundedRectangleBorder(
                borderRadius: BorderRadius.all(Radius.circular(10))),
            padding: EdgeInsets.all(0),
          ),
          child: Container(
            padding: EdgeInsets.only(bottom: 10, top: 18),
            decoration: BoxDecoration(
                color: AppColors.background,
                boxShadow: [
                  BoxShadow(
                      color: Color.fromRGBO(206, 206, 206, 0.05),
                      blurRadius: 5.32,
                      offset: Offset(0, 6.65)),
                  BoxShadow(
                      color: Color.fromRGBO(206, 206, 206, 0.05),
                      blurRadius: 17.87,
                      offset: Offset(0, 22.34)),
                  BoxShadow(
                      color: Color.fromRGBO(206, 206, 206, 0.1),
                      blurRadius: 80,
                      offset: Offset(0, 100))
                ],
                borderRadius: BorderRadius.all(Radius.circular(10))),
```

```
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: <Widget>[
              Padding(
                padding: EdgeInsets.only(left: 18, right: 16),
                child: Text(
                  formatTitle(title),
                  style: TextStyle(
                      fontSize: 16,
                      color: AppColors.black,
                      fontFamily: 'OpenSansBold'),
                ),
              ),
              Padding(
                padding: EdgeInsets.only(left: 12, right: 10),
                child: Html(
                  data: description,
                  style: {
                    "p": Style(
                        fontSize: FontSize(13),
                        color: AppColors.lightBlack,
                        textAlign: TextAlign.justify,
                        fontFamily: 'OpenSansRegular')
                  },
                ),
              ),
              SizedBox(
                height: 3,
              ),
              Padding(
                padding: EdgeInsets.only(left: 18, right: 16),
                child: Row(
                  children: <Widget>[
                    Image.asset(
                      "assets/Calendar.png",
                      width: 18,
                      color: AppColors.blue,
                      height: 18,
                    ),
                    SizedBox(
                      width: 8,
                    ),
                    Text(
                      formatDate(publishDate),
                      style: TextStyle(
                          color: AppColors.blue,
                          fontSize: 11,
                          fontFamily: 'OpenSansBold'),
                    ),
                  ],
                ),
              ),
            ],
          ),
        )),
    );
  }
}
```

Додаток А10. Лістинг коду екрану пошуку

```
import 'package:agrohelper/app/locator.dart';
import 'package:agrohelper/core/model/default_model.dart';
import 'package:agrohelper/ui/shared/my_colors.dart';
import 'package:agrohelper/ui/views/search_view/search_view_model.dart';
import 'package:agrohelper/ui/widgets/items/catalog_item.dart';
import 'package:agrohelper/ui/widgets/my_app_bar.dart';
import 'package:easy_localization/easy_localization.dart';
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:stacked/stacked.dart';

class SearchView extends StatefulWidget {
  @override
  _SearchViewState createState() => _SearchViewState();
}

class _SearchViewState extends State<SearchView> {
  @override
  Widget build(BuildContext context) {
    return ViewModelBuilder<SearchViewModel>.reactive(
        viewModelBuilder: () => locator<SearchViewModel>(),
        onModelReady: (model) => model.init(),
        builder: (context, model, widget) {
          return Scaffold(
            body: Stack(
              fit: StackFit.expand,
              children: <Widget>[
                Image.asset(
                  "assets/background.png",
                  fit: BoxFit.cover,
                ),
                Column(
                  children: <Widget>[
                    MyAppBar(
                      title: tr("search"),
                      expandedHeight: 174,
                    ),
                    Padding(
                      padding: const EdgeInsets.only(left: 15, right: 15),
                      child: Row(
                        children: <Widget>[
                          Expanded(
                            child: Container(
                              height: 48,
                              padding: EdgeInsets.only(left: 10, right: 10),
                              decoration: BoxDecoration(
                                color: Color.fromRGBO(255, 255, 255, 0.5),
                                borderRadius:
                                    BorderRadius.all(Radius.circular(10)),
                              ),
                              child: Row(
                                crossAxisAlignment: CrossAxisAlignment.center,
                                children: <Widget>[
                                  Image.asset(
```

```
                "assets/Search_Glyph.png",
                width: 24,
                height: 24,
                color: Color.fromRGBO(60, 60, 67, 0.6),
              ),
              SizedBox(
                width: 2,
              ),
              Expanded(
                child: TextField(
                  cursorColor: AppColors.darkBlue,
                  cursorWidth: 2,
                  controller: model.textEditingController,
                  style: TextStyle(
                      color: AppColors.darkBlue,
                      fontSize: 17,
                      fontFamily: 'OpenSansRegular'),
                  cursorRadius: Radius.circular(2),
                  decoration: InputDecoration.collapsed(
                    hintText: tr("search"),
                    hintStyle: TextStyle(
                        color: Color.fromRGBO(
                            60, 60, 67, 0.6),
                        fontSize: 17,
                        fontFamily: 'OpenSansRegular'),
                  ),
                ),
              ),
            ],
          )),
    ),
    SizedBox(
      width: 15,
    ),
    TextButton(
      onPressed: model.search,
      style: TextButton.styleFrom(
        shape: RoundedRectangleBorder(
            borderRadius:
                BorderRadius.all(Radius.circular(10))),
        backgroundColor: AppColors.blue,
        padding: EdgeInsets.all(0),
      ),
      child: Container(
        height: 48,
        width: 66,
        child: Center(
          child: Image.asset(
            "assets/Search.png",
            width: 24,
            height: 24,
            color: Colors.white,
          ),
        ),
      ),
    )
  ],
),
```

```
                ),
                SizedBox(
                  height: 5,
                ),
                Expanded(
                  child: AnimatedSwitcher(
                    duration: Duration(milliseconds: 300),
                    child: model.visible
                        ? ListView.builder(
                            itemCount: model.list.length,
                            padding: EdgeInsets.only(
                                right: 15, left: 15, bottom: 80, top: 10),
                            physics: BouncingScrollPhysics(),
                            itemBuilder: (context, index) {
                              DefaultModel item = model.list[index];
                              return CatalogItem(
                                title: item.title,
                                description: item.ko,
                                onPressed: () =>
                                    model.openCatalogItemScreen(item),
                              );
                            },
                          )
                        : Center(
                            child: Container(
                              margin: EdgeInsets.only(bottom: 80),
                              child: Image.asset(
                                "assets/Canister_background.png",
                                width: 205,
                              ),
                            ),
                          ),
                  ),
                ),
              ],
            )
          ],
        ),
      );
    });
  }
}

import 'package:agrohelper/core/model/default_model.dart';
import 'package:agrohelper/core/services/database/database_service.dart';
import 'package:agrohelper/ui/views/catalog/catalog_item_screen.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:injectable/injectable.dart';
import 'package:rxdart/rxdart.dart';
import 'package:stacked/stacked.dart';

@injectable
class SearchViewModel extends BaseViewModel {
  final DatabaseService _databaseService;
  final _searchDataFetcher = PublishSubject<List<dynamic>>();
  final TextEditingController textEditingController = TextEditingController();
```

```
  SearchViewModel(this._databaseService);

  Stream<List<dynamic>> get searchList => _searchDataFetcher.stream;

  BuildContext get context => Get.context;

  bool visible = false;
  List<DefaultModel> list = [];

  init() {
    searchList.listen((list) {
      if (list.isNotEmpty) {
        visible = true;
      } else {
        visible = false;
      }
      this.list = list;
      notifyListeners();
    });
  }

  searchData(String value) async {
    List<dynamic> contactsList = await _databaseService.searchData(value);
    _searchDataFetcher.sink.add(contactsList);
  }

  void search() {
    FocusScope.of(context).unfocus();
    String value = textEditingController.text;
    searchData(value);
  }

  void openCatalogItemScreen(DefaultModel model) {
    Navigator.push(
        context,
        MaterialPageRoute(
            builder: (context) => CatalogItemScreen<DefaultModel>(
                model: model,
              )));
  }

  @override
  void dispose() {
    _searchDataFetcher.close();
    textEditingController.dispose();
    super.dispose();
  }
}
```

Додаток А11. Лістинг коду екрану з насінням

```
import 'package:agrohelper/app/locator.dart';
import 'package:agrohelper/ui/views/seeds/seeds_bloc.dart';
import 'package:agrohelper/core/model/seed_model.dart';
import 'package:agrohelper/core/services/navigation/navigation_service.dart';
import 'package:agrohelper/ui/views/seeds/seeds_details_screen.dart';
import 'package:agrohelper/ui/widgets/items/seeds_item.dart';
```

```
import 'package:agrohelper/ui/widgets/my_sliver_app_bar.dart';
import 'package:agrohelper/ui/widgets/my_sliver_app_header.dart';
import 'package:easy_localization/easy_localization.dart';
import 'package:flutter/material.dart';

class SeedsScreen extends StatefulWidget {
  const SeedsScreen({Key key}) : super(key: key);

  @override
  _SeedsScreenState createState() => _SeedsScreenState();
}

class _SeedsScreenState extends State<SeedsScreen> {
  final SeedsBloc _seedsBloc = SeedsBloc();
  final NavigationService _navigationService = locator<NavigationService>();

  void _onPressed(SeedModel seedModel, {bool isCorn = false}) {
    Navigator.push(
        context,
        MaterialPageRoute(
            builder: (context) => SeedsDetailsScreen(
                seedModel: seedModel,
                isCorn: isCorn,
              )));
  }

  @override
  void initState() {
    super.initState();
    _seedsBloc.fetchSeeds();
  }

  @override
  void dispose() {
    _seedsBloc.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Stack(
      fit: StackFit.expand,
      children: <Widget>[
        Image.asset(
          "assets/background.png",
          fit: BoxFit.cover,
        ),
        NestedScrollView(
          headerSliverBuilder:
              (BuildContext context, bool innerBoxIsScrolled) => <Widget>[
            SliverPersistentHeader(
                pinned: false, delegate: MySliverAppHeader()),
            SliverPersistentHeader(
                pinned: true,
                delegate: MySliverAppBar(
                  onBack: () => _navigationService.pop(),
                  title: tr("seeds"),
                ))
```

```
        ],
        body: StreamBuilder<List<SeedModel>>(
          stream: _seedsBloc.seeds,
          builder: (context, snapshot) {
            List<SeedModel> seeds = snapshot.data ?? [];
            return ListView.builder(
              itemCount: seeds.length,
              padding:
                  EdgeInsets.only(left: 10, right: 10, top: 10, bottom: 80),
              physics: BouncingScrollPhysics(),
              itemBuilder: (context, index) {
                SeedModel seedModel = seeds[index];
                return SeedsItem(
                  title: seedModel.title,
                  description: seedModel.description,
                  imagePath: seedModel.imagePath,
                  onPressed: () {
                    if (index == 0) {
                      _onPressed(seedModel);
                    } else {
                      _onPressed(seedModel, isCorn: true);
                    }
                  },
                );
              },
            );
          }),
      ),
    ],
  );
 }
}

import 'package:agrohelper/ui/shared/my_colors.dart';
import 'package:agrohelper/ui/widgets/items/address_item.dart';
import 'package:agrohelper/ui/widgets/my_sliver_app_bar.dart';
import 'package:agrohelper/ui/widgets/my_sliver_app_header.dart';
import 'package:flutter/material.dart';
import 'package:flutter_html/flutter_html.dart';
import 'package:flutter_html/html_parser.dart';
import 'package:flutter_html/style.dart';

class SeedsItemScreen<T> extends StatefulWidget {
 final T model;

 const SeedsItemScreen({Key key, @required this.model}) : super(key: key);

 @override
 _SeedsItemScreenState createState() => _SeedsItemScreenState();
}

class _SeedsItemScreenState extends State<SeedsItemScreen> {
 Widget drawTable(
   RenderContext context,
   Widget parsedChild,
 ) {
   List<Widget> list = [];
   context.tree.children.forEach((table) {
```

```
    table.children.forEach((tr) {
      if(tr.children.length >= 3) {
        list.add(Container(
          margin: EdgeInsets.only(bottom: 1),
          child: AddressItem(
            isBold: true,
            address: tr.children[0].element.text,
            content: tr.children[2].element.text,
          ),
        ));
      }
    });
  });
  return Container(
    width: MediaQuery.of(this.context).size.width,
    child: Column(
      children: list,
    ),
  );
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: Colors.white,
    body: Stack(
      fit: StackFit.expand,
      children: <Widget>[
        Image.asset(
          "assets/background.png",
          fit: BoxFit.cover,
        ),
        NestedScrollView(
          headerSliverBuilder:
              (BuildContext context, bool innerBoxIsScrolled) => <Widget>[
            SliverPersistentHeader(
                pinned: false, delegate: MySliverAppHeader()),
            SliverPersistentHeader(
                pinned: true,
                delegate: MySliverAppBar(
                  title: widget.model.title,
                ))
          ],
          body: ListView(
            physics: BouncingScrollPhysics(),
            padding: EdgeInsets.only(
              bottom: 80,
              top: 10,
            ),
            children: <Widget>[
              Html(
                data: widget.model.content,
                customRender: {"table": drawTable},
                style: {
                  "div": Style(margin: EdgeInsets.all(0)),
                  "p": Style(
                      fontSize: FontSize(14),
                      color: AppColors.brown,
```

```
                textAlign: TextAlign.justify,
                fontWeight: FontWeight.w500,
                fontFamily: 'OpenSansRegular'),
            "strong": Style(
                fontSize: FontSize(14),
                color: AppColors.brown,
                fontWeight: FontWeight.w500,
                textAlign: TextAlign.justify,
                fontFamily: 'OpenSansRegular')
          },
        ),
      ],
    ),
  ),
 ),
 );
 }
}

import 'package:agrohelper/ui/views/seeds/seeds_bloc.dart';
import 'package:agrohelper/core/model/corn_model.dart';
import 'package:agrohelper/core/model/seed_model.dart';
import 'package:agrohelper/core/model/sunflower_model.dart';
import 'package:agrohelper/ui/views/seeds/seeds_item_screen.dart';
import 'package:agrohelper/ui/widgets/items/corn_item.dart';
import 'package:agrohelper/ui/widgets/items/sunflower_item.dart';
import 'package:agrohelper/ui/widgets/my_sliver_app_bar.dart';
import 'package:agrohelper/ui/widgets/my_sliver_app_header.dart';
import 'package:flutter/material.dart';

class SeedsDetailsScreen extends StatefulWidget {
 final SeedModel seedModel;
 final bool isCorn;

 const SeedsDetailsScreen(
    {Key key, @required this.seedModel, this.isCorn = false})
    : super(key: key);

 @override
 _SeedsDetailsScreenState createState() => _SeedsDetailsScreenState();
}

class _SeedsDetailsScreenState extends State<SeedsDetailsScreen> {
 final SeedsBloc seedsBloc = SeedsBloc();

 void openSeedsItemScreen<T>(T model) {
  Navigator.push(
     context,
     MaterialPageRoute(
        builder: (context) => SeedsItemScreen<T>(
            model: model,
          )));
 }

 @override
 void initState() {
  super.initState();
```

```
  if (widget.isCorn) {
    seedsBloc.fetchCorn();
  } else {
    seedsBloc.fetchSunflowers();
  }
}

@override
void dispose() {
  seedsBloc.dispose();
  super.dispose();
}

StreamBuilder streamBuilder() {
  if (widget.isCorn) {
    return StreamBuilder<List<CornModel>>(
        stream: seedsBloc.corns,
        builder: (context, snapshot) {
          List<CornModel> list = snapshot.data ?? [];
          return ListView.builder(
            itemCount: list.length,
            padding:
                EdgeInsets.only(left: 16, right: 16, bottom: 80, top: 10),
            physics: BouncingScrollPhysics(),
            itemBuilder: (context, index) {
              CornModel cornModel = list[index];
              return CornItem(
                groupOfRipeness: cornModel.groupOfRipeness,
                direction: cornModel.direction,
                title: cornModel.title,
                fao: cornModel.fao,
                typeGrain: cornModel.typeGrain,
                onPressed: () => openSeedsItemScreen<CornModel>(cornModel),
              );
            },
          );
        });
  } else {
    return StreamBuilder<List<SunflowerModel>>(
        stream: seedsBloc.sunflowers,
        builder: (context, snapshot) {
          List<SunflowerModel> list = snapshot.data ?? [];
          return ListView.builder(
            itemCount: list.length,
            padding:
                EdgeInsets.only(left: 16, right: 16, bottom: 80, top: 10),
            physics: BouncingScrollPhysics(),
            itemBuilder: (context, index) {
              SunflowerModel sunflowerModel = list[index];
              return SunflowerItem(
                onPressed: () =>
                    openSeedsItemScreen<SunflowerModel>(sunflowerModel),
                title: sunflowerModel.title,
                direction: sunflowerModel.direction,
                groupOfRipeness: sunflowerModel.groupOfRipeness,
                technology: sunflowerModel.technology,
              );
            },
```

```
          );
        });
    }
  }

  @override
  Widget build(BuildContext context) {
    return Stack(
      fit: StackFit.expand,
      children: <Widget>[
        Image.asset(
          "assets/background.png",
          fit: BoxFit.cover,
        ),
        NestedScrollView(
            headerSliverBuilder:
                (BuildContext context, bool innerBoxIsScrolled) => <Widget>[
                    SliverPersistentHeader(
                        pinned: false, delegate: MySliverAppHeader()),
                    SliverPersistentHeader(
                        pinned: true,
                        delegate: MySliverAppBar(
                          title: widget.seedModel.title,
                        ))
                  ],
            body: streamBuilder()),
      ],
    );
  }
}

import 'package:agrohelper/core/bloc.dart';
import 'package:agrohelper/core/db_repository.dart';
import 'package:agrohelper/core/model/corn_model.dart';
import 'package:agrohelper/core/model/seed_model.dart';
import 'package:agrohelper/core/model/sunflower_model.dart';
import 'package:agrohelper/core/utils/constants.dart';
import 'package:rxdart/rxdart.dart';

class SeedsBloc extends Bloc {
  final repository = DBRepository();
  final _seedFetcher = PublishSubject<List<SeedModel>>();
  final _cornFetcher = PublishSubject<List<CornModel>>();
  final _sunflowerFetcher = PublishSubject<List<SunflowerModel>>();

  Stream<List<SeedModel>> get seeds => _seedFetcher.stream;

  Stream<List<CornModel>> get corns => _cornFetcher.stream;

  Stream<List<SunflowerModel>> get sunflowers => _sunflowerFetcher.stream;

  fetchSeeds() async {
    List<SeedModel> seedsList = await repository.fetchSeeds();
    _seedFetcher.sink.add(seedsList);
  }

  fetchCorn() async {
    CornBox cornsList =
```

```
      await repository.readDefault<CornBox>(Constants.corns, Constants.corns);
    _cornFetcher.sink.add(cornsList.list);
  }

  fetchSunflowers() async {
    SunflowerBox sunflowersList = await repository.readDefault<SunflowerBox>(
        Constants.sunflowers, Constants.sunflowers);
    _sunflowerFetcher.sink.add(sunflowersList.list);
  }

  @override
  void dispose() {
    _cornFetcher.close();
    _sunflowerFetcher.close();
    _seedFetcher.close();
  }
}
```

Додаток А12. Лістинг коду Splash скріну

```
import 'package:agrohelper/app/locator.dart';
import 'package:agrohelper/ui/shared/my_colors.dart';
import 'package:agrohelper/ui/views/splash_view/splash_view_model.dart';
import 'package:easy_localization/easy_localization.dart';
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:stacked/stacked.dart';

class SplashView extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return ViewModelBuilder<SplashViewModel>.reactive(
        viewModelBuilder: () => locator<SplashViewModel>(),
        onModelReady: (model) => model.init(),
        builder: (context, model, widget) {
          return Scaffold(
            key: model.scaffoldKey,
            body: Stack(
              fit: StackFit.expand,
              children: <Widget>[
                Image.asset(
                  "assets/background.png",
                  fit: BoxFit.cover,
                ),
                Center(
                    child: Column(
                  mainAxisSize: MainAxisSize.min,
                  children: <Widget>[
                    Image.asset(
                      "assets/logo.png",
                      fit: BoxFit.cover,
                      height: 115,
                      width: 230,
                    ),
                    Text(
                      tr("splash_text"),
                      textAlign: TextAlign.center,
```

```
                style: TextStyle(
                    fontSize: 12,
                    fontFamily: 'OpenSansRegular',
                    color: AppColors.darkBlue),
              )
            ],
          )),
        ],
      ),
    );
  });
 }
}

import 'package:agrohelper/app/router.dart';
import 'package:agrohelper/core/model/corn_model.dart';
import 'package:agrohelper/core/model/culture_model.dart';
import 'package:agrohelper/core/model/default_model.dart';
import 'package:agrohelper/core/model/sunflower_model.dart';
import 'package:agrohelper/core/model/topic_model.dart';
import 'package:agrohelper/core/services/database/database_service.dart';
import 'package:agrohelper/core/services/navigation/navigation_service.dart';
import 'package:agrohelper/core/services/wordpress/wordpress_service.dart';
import 'package:agrohelper/core/utils/constants.dart';
import 'package:connectivity/connectivity.dart';
import 'package:easy_localization/easy_localization.dart';
import 'package:flutter/material.dart';
import 'package:injectable/injectable.dart';
import 'package:rxdart/subjects.dart';
import 'package:shared_preferences/shared_preferences.dart';
import 'package:stacked/stacked.dart';

@injectable
class SplashViewModel extends BaseViewModel {
 final DatabaseService _databaseService;
 final WordPressService _wordPressService;
 final NavigationService _navigationService;

 final _catalogFetcher = PublishSubject<bool>();
 final GlobalKey<ScaffoldState> scaffoldKey = GlobalKey();

 SplashViewModel(
    this._databaseService, this._wordPressService, this._navigationService);

 Stream<bool> get catalog => _catalogFetcher.stream;

 void init() {
  _isDownload();
 }

 _isDownload() async {
  SharedPreferences prefs = await SharedPreferences.getInstance();
  bool isDownload = prefs.getBool(Constants.isDownload) ?? false;

  catalog.listen((catalog) {
   if (catalog) {
    _setDownload(catalog);
    _openMainScreen();
```

```
    }
   }, onError: (e, stacktrace) {
    print(e);
    print(stacktrace);
    if (isDownload) {
     _openMainScreen();
    } else {
     _showSnackBar();
    }
   });

   var connectivityResult = await (Connectivity().checkConnectivity());
   if (connectivityResult == ConnectivityResult.mobile) {
    _fetchCatalog();
   } else if (connectivityResult == ConnectivityResult.wifi) {
    _fetchCatalog();
   } else if (isDownload) {
    _openMainScreen();
   }
  }

  _showSnackBar() {
   final snackBar = SnackBar(content: Text(tr("download_error")));
   scaffoldKey.currentState.showSnackBar(snackBar);
  }

  _setDownload(bool isDownload) async {
   SharedPreferences prefs = await SharedPreferences.getInstance();
   prefs.setBool(Constants.isDownload, isDownload);
  }

  _openMainScreen() {
   _navigationService.pushNamedAndRemoveUntil(Routes.MainScreen,
     predicate: (Route<dynamic> route) => false);
  }

  _fetchCatalog() async {
   try {
    int version = 1;
    await _databaseService.writeDataBase<int>(
      Constants.version, Constants.version, version);
    List<DefaultModel> catalogList = await _wordPressService.fetchCatalog();
    List<CultureModel> culturesList = await _wordPressService.fetchCulture();
    List<SunflowerModel> sunflowersList =
      await _wordPressService.fetchSunflower();
    List<CornModel> cornsList = await _wordPressService.fetchCorn();
    List<TopicModel> szrTopics = await _wordPressService.fetchSzrTopics();
    List<TopicModel> szrActions = await _wordPressService.fetchSzrActions();
    List<TopicModel> microTopics = await _wordPressService.fetchMicroTopics();

    await _databaseService.writeDataBase<DefaultBox>(
      Constants.catalog, Constants.catalog, DefaultBox(catalogList));

    await _databaseService.writeDataBase<CultureBox>(
      Constants.cultures, Constants.cultures, CultureBox(culturesList));

    await _databaseService.writeDataBase<SunflowerBox>(Constants.sunflowers,
      Constants.sunflowers, SunflowerBox(sunflowersList));
```

```
    await _databaseService.writeDataBase<CornBox>(
      Constants.corns, Constants.corns, CornBox(cornsList));

    await _databaseService.writeDataBase<TopicBox>(
      Constants.szrTopics, "topics", TopicBox(szrTopics));

    await _databaseService.writeDataBase<TopicBox>(
      Constants.szrActions, "topics", TopicBox(szrActions));

    await _databaseService.writeDataBase<TopicBox>(
      Constants.microTopics, "topics", TopicBox(microTopics));

    _catalogFetcher.sink.add(true);
   } catch (e) {
    _catalogFetcher.sink.addError(e);
    // throw e;
   }
  }

  @override
  void dispose() {
   _catalogFetcher.close();
   super.dispose();
  }
}
```

Додаток А13. Лістинг коду кольорів

```
import 'dart:math';
import 'dart:ui';

import 'package:flutter/material.dart';

class AppColors {
  AppColors._();

  static const Color darkGreen = const Color(0xFF29301D);
  static const Color blue = const Color(0xFF4C97D8);
  static const Color brown = const Color(0xFF4F4349);
  static const Color black = const Color(0xFF404353);
  static const Color lightBlack = const Color(0xFF67707B);
  static const Color textBlack = const Color(0xFF7C7C7C);
  static const Color darkBlue = const Color(0xFF005197);
  static const Color background = const Color.fromRGBO(255, 255, 255, 0.75);

  static int _tintValue(int value, double factor) =>
      max(0, min((value + ((255 - value) * factor)).round(), 255));

  static Color _tintColor(Color color, double factor) => Color.fromRGBO(
      _tintValue(color.red, factor),
      _tintValue(color.green, factor),
      _tintValue(color.blue, factor),
      1);

  static int _shadeValue(int value, double factor) =>
      max(0, min(value - (value * factor).round(), 255));
```

```
  static Color _shadeColor(Color color, double factor) => Color.fromRGBO(
      _shadeValue(color.red, factor),
      _shadeValue(color.green, factor),
      _shadeValue(color.blue, factor),
      1);

  static MaterialColor generateMaterialColor(Color color) {
    return MaterialColor(color.value, {
      50: _tintColor(color, 0.9),
      100: _tintColor(color, 0.8),
      200: _tintColor(color, 0.6),
      300: _tintColor(color, 0.4),
      400: _tintColor(color, 0.2),
      500: color,
      600: _shadeColor(color, 0.1),
      700: _shadeColor(color, 0.2),
      800: _shadeColor(color, 0.3),
      900: _shadeColor(color, 0.4),
    });
  }
}
```

Додаток А14. Лістинг коду класів навігації

```
import 'package:agrohelper/ui/views/catalog/catalog_screen.dart';
import 'package:flutter/material.dart';

class TabNavigatorRoutes {
  static const String root = '/';
}

class CatalogTabNavigator extends StatelessWidget {
  CatalogTabNavigator({this.navigatorKey});

  final GlobalKey<NavigatorState> navigatorKey;

  Map<String, WidgetBuilder> _routeBuilders(
    BuildContext context,
  ) {
    return {TabNavigatorRoutes.root: (context) => CatalogScreen()};
  }

  @override
  Widget build(BuildContext context) {
    var routeBuilders = _routeBuilders(context);
    return Navigator(
        key: navigatorKey,
        initialRoute: TabNavigatorRoutes.root,
        onGenerateRoute: (routeSettings) {
          return MaterialPageRoute(
              builder: (context) => routeBuilders[routeSettings.name](context));
        });
  }
}

import 'package:agrohelper/ui/views/contacts/contacts_screen.dart';
```

```
import 'package:flutter/material.dart';

class TabNavigatorRoutes {
  static const String root = '/';
}

class ContactsTabNavigator extends StatelessWidget {
  ContactsTabNavigator({this.navigatorKey});

  final GlobalKey<NavigatorState> navigatorKey;

  Map<String, WidgetBuilder> _routeBuilders(
    BuildContext context,
  ) {
    return {
      TabNavigatorRoutes.root: (context) => ContactsScreen(),
    };
  }

  @override
  Widget build(BuildContext context) {
    var routeBuilders = _routeBuilders(context);

    return Navigator(
        key: navigatorKey,
        initialRoute: TabNavigatorRoutes.root,
        onGenerateRoute: (routeSettings) {
          return MaterialPageRoute(
              builder: (context) => routeBuilders[routeSettings.name](context));
        });
  }
}

import 'package:agrohelper/ui/views/news/news_screen.dart';
import 'package:flutter/material.dart';

class TabNavigatorRoutes {
  static const String root = '/';
}

class NewsTabNavigator extends StatelessWidget {
  NewsTabNavigator({this.navigatorKey});

  final GlobalKey<NavigatorState> navigatorKey;

  Map<String, WidgetBuilder> _routeBuilders(
    BuildContext context,
  ) {
    return {
      TabNavigatorRoutes.root: (context) => NewsScreen(),
    };
  }

  @override
  Widget build(BuildContext context) {
    var routeBuilders = _routeBuilders(context);
    return Navigator(
        key: navigatorKey,
```

```
      initialRoute: TabNavigatorRoutes.root,
      onGenerateRoute: (routeSettings) {
        return MaterialPageRoute(
            builder: (context) => routeBuilders[routeSettings.name](context));
      });
  }
}

import 'package:agrohelper/ui/views/seeds/seeds_screens.dart';
import 'package:flutter/material.dart';

class TabNavigatorRoutes {
  static const String root = '/';
}

class SeedsTabNavigator extends StatelessWidget {
  SeedsTabNavigator({this.navigatorKey});

  final GlobalKey<NavigatorState> navigatorKey;

  Map<String, WidgetBuilder> _routeBuilders(
    BuildContext context,
  ) {
    return {TabNavigatorRoutes.root: (context) => SeedsScreen()};
  }

  @override
  Widget build(BuildContext context) {
    var routeBuilders = _routeBuilders(context);
    return Navigator(
      key: navigatorKey,
      initialRoute: TabNavigatorRoutes.root,
      onGenerateRoute: (routeSettings) {
        return MaterialPageRoute(
            builder: (context) => routeBuilders[routeSettings.name](context));
      });
  }
}
```

Додаток А15. Лістинг класу AboutUsModel

```
class AboutUsModel {
  AboutUsModel({this.content, this.header, this.footer});

  String content;
  bool header;
  bool footer;

  factory AboutUsModel.fromJson(Map<String, dynamic> json) => AboutUsModel(
      content: json["content"],
      header: json["header"] == null ? false : json["header"],
      footer: json["footer"] == null ? false : json["footer"],
    );
}
```

Додаток А16. Лістинг класу ContactModel

```
class ContactModel {
  ContactModel({
    this.city,
    this.addresses,
  });

  String city;
  List<Address> addresses;

  factory ContactModel.fromJson(Map<String, dynamic> json) => ContactModel(
        city: json["city"],
        addresses: List<Address>.from(
            json["addresses"].map((x) => Address.fromJson(x))),
      );
}

class Address {
  Address({
    this.address,
    this.phone,
    this.email,
    this.latitude,
    this.longitude,
  });

  String address;
  List<String> phone;
  String email;
  double latitude;
  double longitude;

  factory Address.fromJson(Map<String, dynamic> json) => Address(
        address: json["address"],
        phone: json["phone"] == null
            ? null
            : List<String>.from(json["phone"].map((x) => x)),
        email: json["email"] == null ? null : json["email"],
        latitude: json["latitude"] == null ? null : json["latitude"].toDouble(),
        longitude:
            json["longitude"] == null ? null : json["longitude"].toDouble(),
      );
}
```

Додаток А17. Лістинг класу Content

```
 class Content {
  Content({
    this.rendered,
    this.protected,
  });

  String rendered;
  bool protected;

  factory Content.fromJson(Map<String, dynamic> json) => Content(
        rendered: json["rendered"],
```

```
      protected: json["protected"],
    );
}
```

Додаток А18. Лістинг класу CornModel

```
import 'package:agrohelper/core/model/title_model.dart';
import 'package:hive/hive.dart';

import 'content_model.dart';

part 'corn_model.g.dart';

@HiveType(typeId: 8)
class CornBox {
  @HiveField(0)
  List<CornModel> list;

  CornBox(this.list);
}

@HiveType(typeId: 3)
class CornModel {
  CornModel({
    this.id,
    this.title,
    this.content,
    this.fao,
    this.groupOfRipeness,
    this.typeGrain,
    this.direction,
  });

  @HiveField(0)
  int id;
  @HiveField(1)
  String title;
  @HiveField(2)
  String content;
  @HiveField(3)
  String fao;
  @HiveField(4)
  String groupOfRipeness;
  @HiveField(5)
  String typeGrain;
  @HiveField(6)
  String direction;

  factory CornModel.fromJson(Map<String, dynamic> json) => CornModel(
      id: json["id"],
      title: Title.fromJson(json["title"]).rendered,
      content: Content.fromJson(json["content"]).rendered,
      fao: json["fao"],
      groupOfRipeness: json["group_of_ripeness"],
      typeGrain: json["type_grain"],
      direction: json["direction"],
    );
```

```
}
```

Додаток А19. Лістинг класу CultureModel

```
import 'package:hive/hive.dart';

import 'content_model.dart';
import 'title_model.dart';

part 'culture_model.g.dart';

@HiveType(typeId: 7)
class CultureBox {
  @HiveField(0)
  List<CultureModel> list;

  CultureBox(this.list);
}

@HiveType(typeId: 1)
class CultureModel {
  CultureModel(
      {this.id,
      this.title,
      this.content,
      this.topics,
      this.typesMicro,
      this.userProizv,
      this.decriptionCustom});

  @HiveField(0)
  int id;
  @HiveField(1)
  String title;
  @HiveField(2)
  String content;
  @HiveField(3)
  List<int> topics;
  @HiveField(4)
  List<int> typesMicro;
  @HiveField(5)
  String userProizv;
  @HiveField(6)
  String decriptionCustom;

  factory CultureModel.fromJson(Map<String, dynamic> json) => CultureModel(
        id: json["id"],
        title: Title.fromJson(json["title"]).rendered,
        content: Content.fromJson(json["content"]).rendered,
        topics: json["topics"] != null
            ? List<int>.from(json["topics"].map((x) => x))
            : null,
        typesMicro: json["types_micro"] != null
            ? List<int>.from(json["types_micro"].map((x) => x))
            : null,
        decriptionCustom: json["decription_custom"],
        userProizv: json["user_proizv"],
```

```
    );
}
```

Додаток А20. Лістинг класу DefaultModel

```
import 'package:agrohelper/core/model/title_model.dart';
import 'package:hive/hive.dart';

import 'content_model.dart';

part 'default_model.g.dart';

@HiveType(typeId: 9)
class DefaultBox {
  @HiveField(0)
  List<DefaultModel> list;

  DefaultBox(this.list);
}

@HiveType(typeId: 0)
class DefaultModel {
  DefaultModel({
    this.id,
    this.title,
    this.content,
    this.szrTopics,
    this.szrTypes,
    this.szrActions,
    this.ko,
  });

  @HiveField(0)
  int id;
  @HiveField(1)
  String title;
  @HiveField(2)
  String content;
  @HiveField(3)
  List<int> szrTopics;
  @HiveField(4)
  List<int> szrTypes;
  @HiveField(5)
  List<int> szrActions;
  @HiveField(6)
  String ko;

  factory DefaultModel.fromJson(Map<String, dynamic> json) => DefaultModel(
      id: json["id"],
      title: Title.fromJson(json["title"]).rendered,
      content: Content.fromJson(json["content"]).rendered,
      szrTopics: List<int>.from(json["szr_topics"].map((x) => x)),
      szrTypes: List<int>.from(json["szr_types"].map((x) => x)),
      szrActions: List<int>.from(json["szr_actions"].map((x) => x)),
      ko: json["ko"],
    );
}
```

Додаток А21. Лістинг класу NewsModel

```
import 'content_model.dart';
import 'title_model.dart';

class NewsModel {
  NewsModel({
    this.id,
    this.date,
    this.dateGmt,
    this.modified,
    this.modifiedGmt,
    this.slug,
    this.status,
    this.type,
    this.link,
    this.title,
    this.content,
    this.excerpt,
  });

  int id;
  DateTime date;
  DateTime dateGmt;
  DateTime modified;
  DateTime modifiedGmt;
  String slug;
  String status;
  String type;
  String link;
  Title title;
  Content content;
  Content excerpt;

  factory NewsModel.fromJson(Map<String, dynamic> json) => NewsModel(
        id: json["id"],
        date: DateTime.parse(json["date"]),
        dateGmt: DateTime.parse(json["date_gmt"]),
        modified: DateTime.parse(json["modified"]),
        modifiedGmt: DateTime.parse(json["modified_gmt"]),
        slug: json["slug"],
        status: json["status"],
        type: json["type"],
        link: json["link"],
        title: Title.fromJson(json["title"]),
        content: Content.fromJson(json["content"]),
        excerpt: Content.fromJson(json["excerpt"]),
      );
}
```

Додаток А22. Лістинг класу SeedModel

```
class SeedModel {
  SeedModel({
```

```
    this.title,
    this.description,
    this.imagePath,
  });

  String title;
  String description;
  String imagePath;

  factory SeedModel.fromJson(Map<String, dynamic> json) => SeedModel(
        title: json["title"],
        description: json["description"],
        imagePath: "assets/" + json["imagePath"],
      );
}
```

Додаток А23. Лістинг класу SunflowerModel

```
import 'package:hive/hive.dart';

import 'content_model.dart';
import 'title_model.dart';

part 'sunflower_model.g.dart';

@HiveType(typeId: 5)
class SunflowerBox {
  @HiveField(0)
  List<SunflowerModel> list;

  SunflowerBox(this.list);
}

@HiveType(typeId: 2)
class SunflowerModel {
  SunflowerModel({
    this.id,
    this.title,
    this.content,
    this.groupOfRipeness,
    this.technology,
    this.direction,
  });

  @HiveField(0)
  int id;
  @HiveField(1)
  String title;
  @HiveField(2)
  String content;
  @HiveField(3)
  String groupOfRipeness;
  @HiveField(4)
  String technology;
  @HiveField(5)
  String direction;
```

```
  factory SunflowerModel.fromJson(Map<String, dynamic> json) => SunflowerModel(
        id: json["id"],
        title: Title.fromJson(json["title"]).rendered,
        content: Content.fromJson(json["content"]).rendered,
        groupOfRipeness: json["group_of_ripeness"],
        technology: json["technology"],
        direction: json["direction"],
      );
}
```

Додаток А24. Лістинг класу Title

```
class Title {
  Title({
    this.rendered,
  });

  String rendered;

  factory Title.fromJson(Map<String, dynamic> json) => Title(
        rendered: json["rendered"],
      );
}
```

Додаток А25. Лістинг класу TopicModel

```
import 'package:hive/hive.dart';

part 'topic_model.g.dart';

@HiveType(typeId: 6)
class TopicBox {
  @HiveField(0)
  List<TopicModel> list;

  TopicBox(this.list);
}

//flutter packages pub run build_runner build
@HiveType(typeId: 4)
class TopicModel {
  TopicModel({
    this.id,
    this.name,
  });

  @HiveField(0)
  int id;
  @HiveField(1)
  String name;

  factory TopicModel.fromJson(Map<String, dynamic> json) => TopicModel(
        id: json["id"],
        name: json["name"],
      );
}
```

Додаток А26. Лістинг класу WeatherModel

```
class WeatherModel {
  WeatherModel({
    this.lat,
    this.lon,
    this.timezone,
    this.timezoneOffset,
    this.daily,
  });

  double lat;
  double lon;
  String timezone;
  int timezoneOffset;
  List<Daily> daily;

  factory WeatherModel.fromJson(Map<String, dynamic> json) => WeatherModel(
        lat: json["lat"].toDouble(),
        lon: json["lon"].toDouble(),
        timezone: json["timezone"],
        timezoneOffset: json["timezone_offset"],
        daily: List<Daily>.from(json["daily"].map((x) => Daily.fromJson(x))),
      );
}

class Daily {
  Daily({
    this.dt,
    this.sunrise,
    this.sunset,
    this.temp,
    this.feelsLike,
    this.pressure,
    this.humidity,
    this.dewPoint,
    this.windSpeed,
    this.windDeg,
    this.weather,
    this.clouds,
    this.uvi,
    this.rain,
  });

  int dt;
  int sunrise;
  int sunset;
  Temp temp;
  FeelsLike feelsLike;
  int pressure;
  int humidity;
  double dewPoint;
  double windSpeed;
  int windDeg;
  List<Weather> weather;
  int clouds;
```

```
  double uvi;
  double rain;

  factory Daily.fromJson(Map<String, dynamic> json) => Daily(
      dt: json["dt"],
      sunrise: json["sunrise"],
      sunset: json["sunset"],
      temp: Temp.fromJson(json["temp"]),
      feelsLike: FeelsLike.fromJson(json["feels_like"]),
      pressure: json["pressure"],
      humidity: json["humidity"],
      dewPoint: json["dew_point"].toDouble(),
      windSpeed: json["wind_speed"].toDouble(),
      windDeg: json["wind_deg"],
      weather:
         List<Weather>.from(json["weather"].map((x) => Weather.fromJson(x))),
      clouds: json["clouds"],
      uvi: json["uvi"].toDouble(),
      rain: json["rain"] == null ? null : json["rain"].toDouble(),
     );
}

class FeelsLike {
  FeelsLike({
    this.day,
    this.night,
    this.eve,
    this.morn,
  });

  double day;
  double night;
  double eve;
  double morn;

  factory FeelsLike.fromJson(Map<String, dynamic> json) => FeelsLike(
      day: json["day"].toDouble(),
      night: json["night"].toDouble(),
      eve: json["eve"].toDouble(),
      morn: json["morn"].toDouble(),
     );
}

class Temp {
  Temp({
    this.day,
    this.min,
    this.max,
    this.night,
    this.eve,
    this.morn,
  });

  double day;
  double min;
  double max;
  double night;
  double eve;
```

```
  double morn;

  factory Temp.fromJson(Map<String, dynamic> json) => Temp(
        day: json["day"].toDouble(),
        min: json["min"].toDouble(),
        max: json["max"].toDouble(),
        night: json["night"].toDouble(),
        eve: json["eve"].toDouble(),
        morn: json["morn"].toDouble(),
      );
}

class Weather {
  Weather({
    this.id,
    this.main,
    this.description,
    this.icon,
  });

  int id;
  MainEnum main;
  Description description;
  String icon;

  factory Weather.fromJson(Map<String, dynamic> json) => Weather(
        id: json["id"],
        main: mainEnumValues.map[json["main"]],
        description: descriptionValues.map[json["description"]],
        icon: json["icon"],
      );
}

enum Description {
  BROKEN_CLOUDS,
  CLEAR_SKY,
  OVERCAST_CLOUDS,
  LIGHT_SNOW,
  FEW_CLOUDS,
  LIGHT_RAIN,
  SCATTERED_CLOUDS
}

final descriptionValues = EnumValues({
  "broken clouds": Description.BROKEN_CLOUDS,
  "clear sky": Description.CLEAR_SKY,
  "few clouds": Description.FEW_CLOUDS,
  "light rain": Description.LIGHT_RAIN,
  "light snow": Description.LIGHT_SNOW,
  "overcast clouds": Description.OVERCAST_CLOUDS,
  "scattered clouds": Description.SCATTERED_CLOUDS
});

enum MainEnum { CLOUDS, CLEAR, SNOW, RAIN }

final mainEnumValues = EnumValues({
  "Clear": MainEnum.CLEAR,
  "Clouds": MainEnum.CLOUDS,
```

```
  "Rain": MainEnum.RAIN,
  "Snow": MainEnum.SNOW
});

class EnumValues<T> {
  Map<String, T> map;
  Map<T, String> reverseMap;

  EnumValues(this.map);

  Map<T, String> get reverse {
    if (reverseMap == null) {
      reverseMap = map.map((k, v) => new MapEntry(v, k));
    }
    return reverseMap;
  }
}
```

Додаток А27. Лістинг коди сервісу бази даних

```
import 'package:agrohelper/core/model/about_us_model.dart';
import 'package:agrohelper/core/model/contacts_model.dart';
import 'package:agrohelper/core/model/culture_model.dart';
import 'package:agrohelper/core/model/default_model.dart';
import 'package:agrohelper/core/model/seed_model.dart';

abstract class DatabaseService {
  Future<bool> writeDataBase<T>(String key, String boxName, T value);

  Future<List<DefaultModel>> searchData(String value);

  Future<List<DefaultModel>> readCatalog(
      int catalogId, int szrTopics, int szrActions);

  Future<List<CultureModel>> readCultures(int microTopics);

  Future<int> readVersion();

  Future<List<SeedModel>> fetchSeeds();

  Future<List<AboutUsModel>> fetchAbout();

  Future<T> readDefault<T>(String key, String boxName);

  Future<List<ContactModel>> fetchContacts();
}

import 'dart:convert';

import 'package:agrohelper/core/model/about_us_model.dart';
import 'package:agrohelper/core/model/contacts_model.dart';
import 'package:agrohelper/core/model/culture_model.dart';
import 'package:agrohelper/core/model/default_model.dart';
import 'package:agrohelper/core/model/seed_model.dart';
import 'package:agrohelper/core/utils/constants.dart';
import 'package:flutter/services.dart' show rootBundle;
import 'package:hive/hive.dart';
```

```
import 'package:injectable/injectable.dart';

import 'database_service.dart';

@Injectable(as: DatabaseService)
class DatabaseServiceImpl extends DatabaseService {
 @override
 Future<bool> writeDataBase<T>(String key, String boxName, T value) async {
  try {
   Box<T> box = await Hive.openBox<T>(boxName);
   box.put(key, value);
   return true;
  } catch (e) {
   throw e;
  }
 }

 @override
 Future<List<DefaultModel>> searchData(String value) async {
  try {
   Box<DefaultBox> catalogBox =
     await Hive.openBox<DefaultBox>(Constants.catalog);
   Box<CultureBox> culturesBox =
     await Hive.openBox<CultureBox>(Constants.cultures);
   List<DefaultModel> catalogList = catalogBox.get(Constants.catalog).list;
   List<CultureModel> culturesList =
     culturesBox.get(Constants.cultures).list;
   List<DefaultModel> culturesDList = culturesList
     .map((e) => DefaultModel(
       id: e.id, title: e.title, content: e.content, ko: e.userProizv))
     .toList();
   List<DefaultModel> mainList = [];
   mainList.addAll(catalogList);
   mainList.addAll(culturesDList);
   mainList = mainList.where((element) {
    String title = element.title;
    String content = element.content;
    return title.toLowerCase().contains(value.toLowerCase()) ||
      content.toLowerCase().contains(value.toLowerCase());
   }).toList();
   return mainList;
  } catch (e) {
   throw e;
  }
 }

 @override
 Future<List<DefaultModel>> readCatalog(
   int catalogId, int szrTopics, int szrActions) async {
  try {
   Box<DefaultBox> box = await Hive.openBox<DefaultBox>(Constants.catalog);
   List<DefaultModel> mainList = box.get(Constants.catalog).list;

   mainList = mainList
     .where((element) => element.szrTypes.contains(catalogId))
     .toList();
   if (szrTopics != null) {
    mainList = mainList
```

```
        .where((element) => element.szrTopics.contains(szrTopics))
        .toList();
    }
    if (szrActions != null) {
      mainList = mainList
        .where((element) => element.szrActions.contains(szrActions))
        .toList();
    }
    return mainList;
  } catch (e) {
    throw e;
  }
}

@override
Future<List<CultureModel>> readCultures(int microTopics) async {
  try {
    Box<CultureBox> box = await Hive.openBox<CultureBox>(Constants.cultures);
    List<CultureModel> mainList = box.get(Constants.cultures).list;
    if (microTopics != null) {
      mainList = mainList
        .where((element) => element.topics.contains(microTopics))
        .toList();
    }
    return mainList;
  } catch (e) {
    throw e;
  }
}

@override
Future<T> readDefault<T>(String key, String boxName) async {
  try {
    var box = await Hive.openBox<T>(boxName);
    var boxModel = box.get(key);
    return boxModel;
  } catch (e) {
    throw e;
  }
}

@override
Future<int> readVersion() async {
  try {
    Box<int> box = await Hive.openBox<int>(Constants.version);
    int version = box.get(Constants.version);
    return version;
  } catch (e) {
    throw e;
  }
}

@override
Future<List<ContactModel>> fetchContacts() async {
  try {
    String fileName = "contacts.json";
    String data = await rootBundle.loadString('assets/json/$fileName');
    return List<ContactModel>.from(
```

```
        json.decode(data).map((x) => ContactModel.fromJson(x)));
    } catch (e) {
      throw e;
    }
  }

  @override
  Future<List<SeedModel>> fetchSeeds() async {
    try {
      String data = await rootBundle.loadString('assets/json/seeds.json');
      return List<SeedModel>.from(
          json.decode(data).map((x) => SeedModel.fromJson(x)));
    } catch (e) {
      throw e;
    }
  }

  @override
  Future<List<AboutUsModel>> fetchAbout() async {
    try {
      String data = await rootBundle.loadString('assets/json/about_us.json');
      return List<AboutUsModel>.from(
          json.decode(data).map((x) => AboutUsModel.fromJson(x)));
    } catch (e) {
      throw e;
    }
  }
}

class DBRepository {
  final DatabaseService _databaseService = locator<DatabaseService>();

  Future<List<ContactModel>> fetchContacts() =>
      _databaseService.fetchContacts();

  Future<List<SeedModel>> fetchSeeds() => _databaseService.fetchSeeds();

  Future<List<AboutUsModel>> fetchAbout() async =>
      _databaseService.fetchAbout();

  Future<int> readVersion() => _databaseService.readVersion();

  Future<bool> writeDataBase<T>(String boxName, String key, T value) =>
      _databaseService.writeDataBase<T>(boxName, key, value);

  Future<List<DefaultModel>> readCatalog(
          int catalogId, int szrTopics, int szrActions) =>
      _databaseService.readCatalog(catalogId, szrTopics, szrActions);

  Future<List<dynamic>> searchData(String value) =>
      _databaseService.searchData(value);

  Future<List<CultureModel>> readCultures(int microTopics) =>
      _databaseService.readCultures(microTopics);

  Future<T> readDefault<T>(String key, String boxName) =>
      _databaseService.readDefault<T>(key, boxName);
}
```

Додаток А28. Лістинг коду сервісу навігації

```
import 'package:flutter/widgets.dart'
    show GlobalKey, NavigatorState, RoutePredicate, Widget, required;

/// Service that is responsible for navigating around the app
abstract class NavigationService {
  GlobalKey<NavigatorState> get navigatorKey;

  Future<dynamic> pushNamed(String routeName, {dynamic arguments});

  Future<dynamic> push(Widget widget, {dynamic arguments});

  Future<dynamic> pushReplacementNamed(String routeName, {dynamic arguments});

  Future<dynamic> popAllAndPushNamed(String routeName, {dynamic arguments});

  Future<dynamic> popUntilFirstAndPushNamed(String routeName,
      {dynamic arguments});

  Future<dynamic> pushNamedAndRemoveUntil(String routeName,
      {@required RoutePredicate predicate, arguments, int id});

  void popUntilNamed(String routeName);

  void popUntilPredicate(RoutePredicate routePredicate);

  void popRepeated(int times);

  bool pop({dynamic returnValue});
}

import 'package:flutter/material.dart';
import 'package:flutter/widgets.dart' show GlobalKey, NavigatorState;
import 'package:get/get.dart';
import 'package:injectable/injectable.dart';

import 'navigation_service.dart';

@LazySingleton(as: NavigationService)
class NavigationServiceImpl implements NavigationService {
  @override
  GlobalKey<NavigatorState> get navigatorKey => Get.key;

  /// Pushes [routeName] onto the navigation stack
  @override
  Future<dynamic> pushNamed(String routeName, {dynamic arguments}) {
    return Get.toNamed(routeName, arguments: arguments);
  }

  /// Replaces the current route with the [routeName]
  @override
  Future<dynamic> pushReplacementNamed(String routeName, {dynamic arguments}) {
    return Get.offNamed(
      routeName,
      arguments: arguments,
```

```
  );
}

/// Pops the current scope and indicates if you can pop again
@override
bool pop({dynamic returnValue}) {
  Get.back(result: returnValue);
  return Get.key.currentState.canPop();
}

/// Pops the back stack the number of times you indicate with [popTimes]
void popRepeated(int popTimes) {
  Get.close(popTimes);
}

/// Pops the back stack to a route name
@override
void popUntilNamed(String routeName) {
  Get.until(ModalRoute.withName(routeName));
}

/// Pops the back stack until the predicate is satisfied
@override
void popUntilPredicate(RoutePredicate predicate) {
  Get.until(predicate);
}

/// Clears the entire back stack and shows [routeName]
@override
Future<dynamic> popAllAndPushNamed(String routeName, {dynamic arguments}) {
  return Get.offAllNamed(routeName, arguments: arguments);
}

/// Pops the navigation stack until there's 1 view left then pushes [routeName] onto the stack
@override
Future<dynamic> popUntilFirstAndPushNamed(String routeName,
    {dynamic arguments}) {
  _clearBackstackTillFirst();

  return pushNamed(routeName, arguments: arguments);
}

/// Push route and clear stack until predicate is satisfied
@override
Future<dynamic> pushNamedAndRemoveUntil(String routeName,
    {@required RoutePredicate predicate, arguments, int id}) {
  return Get.offAllNamed(routeName,
      predicate: predicate, arguments: arguments, id: id);
}

void _clearBackstackTillFirst() {
  Get.until((Route route) => route.isFirst);
}

@override
Future push(Widget widget, {arguments}) {
  return Get.to(() => widget, arguments: arguments);
}
```

```
}
```

Додаток А29. Лістинг коду сервісу погоди

```
import 'package:agrohelper/core/model/weather_model.dart';

abstract class WeatherService {
  Future<WeatherModel> fetchWeather(double lat, double lon);
}

import 'dart:convert';

import 'package:agrohelper/core/model/weather_model.dart';
import 'package:agrohelper/core/utils/constants.dart';
import 'package:http/http.dart';
import 'package:injectable/injectable.dart';

import 'weather_service.dart';

@Injectable(as: WeatherService)
class WeatherServiceImpl extends WeatherService {
  Client client = Client();

  @override
  Future<WeatherModel> fetchWeather(double lat, double lon) async {
    final appid = Constants.appid;
    final response = await client
        .get(Uri.parse(

"https://api.openweathermap.org/data/2.5/onecall?lat=$lat&lon=$lon&exclude=current,hourly,minutely&appid=
$appid&units=metric"))
        .timeout(const Duration(seconds: 8))
        .catchError((e) {
      throw Exception('Failed to load post');
    });
    if (response.statusCode == 200) {
      return WeatherModel.fromJson(json.decode(response.body));
    } else {
      throw Exception('Failed to load post');
    }
  }
}
```

Додаток А30. Лістинг коду сервісу WordPress

```
import 'package:agrohelper/core/model/corn_model.dart';
import 'package:agrohelper/core/model/culture_model.dart';
import 'package:agrohelper/core/model/default_model.dart';
import 'package:agrohelper/core/model/news_model.dart';
import 'package:agrohelper/core/model/sunflower_model.dart';
import 'package:agrohelper/core/model/topic_model.dart';

abstract class WordPressService {
  Future<List<DefaultModel>> fetchCatalog();

  Future<List<CornModel>> fetchCorn();
```

```
  Future<List<SunflowerModel>> fetchSunflower();

  Future<List<CultureModel>> fetchCulture();

  Future<List<TopicModel>> fetchSzrTopics();

  Future<List<TopicModel>> fetchMicroTopics();

  Future<List<TopicModel>> fetchSzrActions();

  Future<List<NewsModel>> fetchNews();
}

import 'package:agrohelper/core/model/corn_model.dart';
import 'package:agrohelper/core/model/culture_model.dart';
import 'package:agrohelper/core/model/default_model.dart';
import 'package:agrohelper/core/model/news_model.dart';
import 'package:agrohelper/core/model/sunflower_model.dart';
import 'package:agrohelper/core/model/topic_model.dart';
import 'package:agrohelper/core/services/wordpress/wordpress_service.dart';
import 'package:agrohelper/core/utils/constants.dart';
import 'package:dio/dio.dart';
import 'package:injectable/injectable.dart';

@Injectable(as: WordPressService)
class WordPressServiceImpl extends WordPressService {
  static BaseOptions options = BaseOptions(
      baseUrl: Constants.url, receiveTimeout: 15000, connectTimeout: 15000);

  Dio dio = Dio(options);

  @override
  Future<List<DefaultModel>> fetchCatalog() async {
    final response = await dio
        .get("szrs", queryParameters: {"per_page": "100", "lang": "uk"});
    if (response.statusCode == 200) {
      return List<DefaultModel>.from(
          response.data.map((x) => DefaultModel.fromJson(x)));
    } else {
      throw Exception('Failed to load post');
    }
  }

  @override
  Future<List<CornModel>> fetchCorn() async {
    final response = await dio
        .get("corn", queryParameters: {"per_page": "100", "lang": "uk"});
    if (response.statusCode == 200) {
      return List<CornModel>.from(
          response.data.map((x) => CornModel.fromJson(x)));
    } else {
      throw Exception('Failed to load post');
    }
  }

  @override
  Future<List<SunflowerModel>> fetchSunflower() async {
```

```
  final response = await dio
      .get("sunflower", queryParameters: {"per_page": "100", "lang": "uk"});
  if (response.statusCode == 200) {
    return List<SunflowerModel>.from(
        response.data.map((x) => SunflowerModel.fromJson(x)));
  } else {
    throw Exception('Failed to load post');
  }
}

@override
Future<List<CultureModel>> fetchCulture() async {
  final response = await dio
      .get("culture", queryParameters: {"per_page": "100", "lang": "uk"});
  if (response.statusCode == 200) {
    return List<CultureModel>.from(
        response.data.map((x) => CultureModel.fromJson(x)));
  } else {
    throw Exception('Failed to load post');
  }
}

@override
Future<List<TopicModel>> fetchSzrTopics() async {
  final response = await dio
      .get("szr_topics", queryParameters: {"per_page": "100", "lang": "uk"});
  if (response.statusCode == 200) {
    return List<TopicModel>.from(
        response.data.map((x) => TopicModel.fromJson(x)));
  } else {
    throw Exception('Failed to load post');
  }
}
@override
Future<List<TopicModel>> fetchMicroTopics() async {
  final response = await dio
      .get("topics", queryParameters: {"per_page": "100", "lang": "uk"});
  if (response.statusCode == 200) {
    return List<TopicModel>.from(
        response.data.map((x) => TopicModel.fromJson(x)));
  } else {
    throw Exception('Failed to load post');
  }
}

@override
Future<List<TopicModel>> fetchSzrActions() async {
  final response = await dio
      .get("szr_actions", queryParameters: {"per_page": "100", "lang": "uk"});
  if (response.statusCode == 200) {
    return List<TopicModel>.from(
        response.data.map((x) => TopicModel.fromJson(x)));
  } else {
    throw Exception('Failed to load post');
  }
}

@override
```

```
  Future<List<NewsModel>> fetchNews() async {
    final response = await dio
        .get("posts", queryParameters: {"per_page": "100", "lang": "uk"});
    if (response.statusCode == 200) {
      return List<NewsModel>.from(
          response.data.map((x) => NewsModel.fromJson(x)));
    } else {
      throw Exception('Failed to load post');
    }
  }
}
```

Додаток А31. Лістинг коду констант

```
class Constants {
  static const String catalog = "catalog";
  static const String cultures = "cultures";
  static const String version = "version";
  static const String sunflowers = "sunflowers";
  static const String corns = "corns";
  static const String topics = "topics";

  static const double defaultLatitude = 50.4501;
  static const double defaultLongitude = 30.5234;

  static const String szrTopics = "szrTopics";
  static const String szrActions = "szrActions";
  static const String microTopics = "microTopics";

  static const String isDownload = "isDownload";
  static const String appid = "5bf8fd1351ab62bab6a798464615f557";

  static const int gerbicidy = 251;
  static const int desikanty = 305;
  static const int insekticidy = 300;
  static const int protraviteli = 309;
  static const int rodenticidy = 159;
  static const int smachivatel = 324;
  static const int fumiganty = 238;
  static const int fungicidy = 287;

  static String url = "http://10.0.2.2/wp-json/wp/v2/";
}

abstract class Bloc {
  //Need to implement dispose of all Bloc classes
  void dispose();
}
```