

Kierunek: **INF**
Specjalność:

PRACA DYPLOMOWA
INŻYNIERSKA

Aplikacja mobilna wspomagająca pielęgnację roślin
doniczkowych

Małgorzata Skrok

Opiekun pracy
Dr inż. Michał Szczepanik

Słowa kluczowe: aplikacja mobilna, system Android, rośliny doniczkowe

Streszczenie

Celem niniejszej pracy jest zaprojektowanie i implementacja aplikacji na urządzenia mobilne z systemem Android. Aplikacja ta ma za zadanie ułatwić użytkownikom opiekę nad roślinami doniczkowymi poprzez możliwość ustawiania różnego rodzaju przypomnień, zapewnianie informacji na temat istniejących gatunków i odmian, oraz wspomóc w zarządzaniu ich domową kolekcją poprzez listę życzeń. Największą słabość istniejących aplikacji stanowi mała elastyczność dostępnych rodzajów przypomnień oraz brak wsparcia użytkownika w zakresie wiedzy. Dodatkowo żadna z nich nie pozwala na przechowywanie listy roślin, które użytkownik chciałby dodać do swojej kolekcji. Zaproponowane rozwiązanie oferuje liczne możliwości w zakresie reprezentacji domowych roślin, elastyczne przypomnienia, wsparcie wiedzy w postaci biblioteki gatunków i odmian oraz listę życzeń. Aplikacja została napisana w przeważającej mierze w języku Kotlin, a w drodze implementacji wykorzystane zostały takie narzędzia jak Room, ProGuard oraz Koin. Niniejsza praca zawiera deskrypcje konkurencyjnych rozwiązań, projekt aplikacji oraz opisy implementacji i testów.

Abstract

The purpose of the study is to design and implement an application intended for mobile devices with Android operating system. This application is meant to make it easier for its' users to take care of their indoor plants by allowing them to set different kinds of alarms, providing them with necessary information about different species and varieties, and by allowing the user to create a personal wish list to keep track of their growing collection. The biggest weakness of existing solutions is the lack of flexibility in terms of different kinds of alarms and insufficient support in terms of access to information. Additionally none of them allow their users to keep a wish list of plants which they would like to add to their collection. The application has been written mostly in Kotlin programming language, with assistance of tools like Room, ProGuard and Koin. The following study includes descriptions of competitive solutions, an overview of the design process, implementation and testing.

1. Spis treści

Spis treści

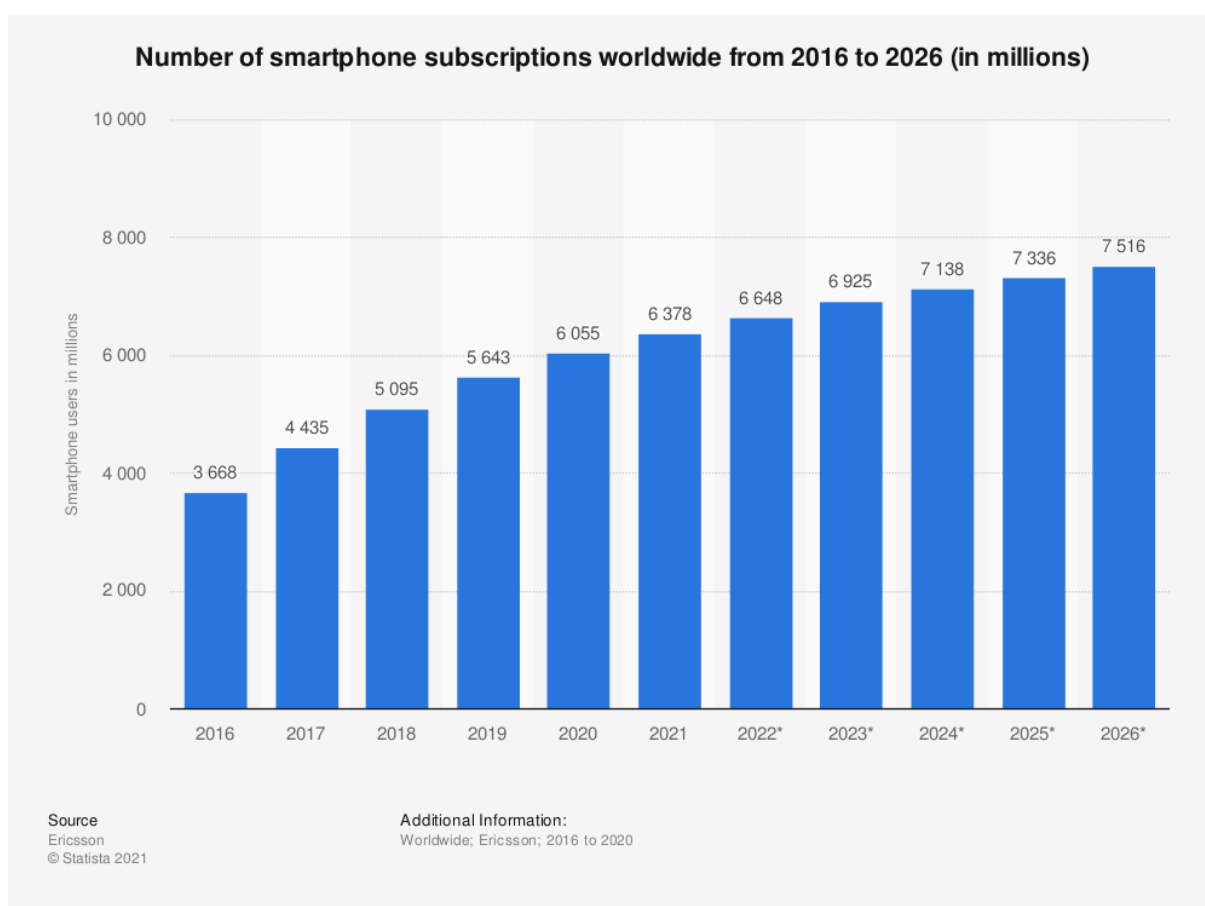
Streszczenie	2
Abstract	2
1. Spis treści	3
2. Wstęp	6
3. Cel i zakres pracy	8
3.1. Analiza istniejących rozwiązań	8
3.2. Opis technologii	8
3.3. Projekt Aplikacji	8
3.4. Implementacja	8
4. Analiza istniejących rozwiązań	9
4.1. Wstęp	9
4.1. Waterbot	9
4.2. Planteo	9
4.3. Garden Manager	10
4.4. Plant Care Reminder	10
4.5. Vera	11
4.6. Smart doniczka – Xiaomi RoPot	12
4.7. Podsumowanie	12
5. Wykorzystane technologie	13
5.1. Wstęp	13
5.2. System Android	13
5.3. Git	14
5.4. Java	15
5.5. Kotlin	16
5.6. Material Design	16
5.7. Firebase	17
5.8. Android Studio	17
5.9. Android Architecture Components	17
5.9.1. Live Data	17
5.9.2. ViewModel	18
5.9.3. Room	18
5.9.4. Data Binding	18
5.9.5. Navigation	18

5.10.	Koin	19
5.11.	Biblioteki zewnętrzne – podsumowanie	19
6.	Zarys projektu	20
6.1.	Wstęp	20
6.1.	Wymagania	20
6.1.1.	Wymagania funkcjonalne	20
6.1.2.	Wymagania нефункционалне	20
6.2.	Przypadki użycia	21
6.2.1.	Diagram przypadków użycia	22
6.2.2.	Scenariusze przypadków użycia	23
6.3.	Baza danych	37
6.3.1.	Room	37
6.3.2.	Relacje	37
6.3.3.	Schemat bazy danych	38
6.4.	Diagram Klas	39
6.4.1.	Diagram klas pakietu data	40
6.4.2.	Diagram klas pakietu domain	41
6.4.3.	Diagram klas pakietu presentation	42
6.5.	Projekt interfejsu	43
7.	Implementacja	45
7.1.	Wstęp	45
7.2.	Środowisko	45
7.3.	Elementy projektu	45
7.4.	Wzorce projektowe	47
7.4.1.	Adapter	47
7.4.2.	Mediator	47
7.4.3.	Wstrzykiwanie zależności	47
7.5.	Implementacja bazy danych	48
7.6.	Problemy implementacyjne	49
7.7.	Testy	50
7.7.1.	Testy jednostkowe	50
7.7.2.	Testy integracyjne	51
7.8.	Interfejs użytkownika	51
7.8.1.	Ekran logowania	51
7.8.2.	Ekrany dolnego paska aplikacji	52
7.8.3.	Ekrany dodawania/edycji	53

7.8.4.	Ekrany szczegółów obiektów	53
8.	Podsumowanie	54
9.	Bibliografia.....	55
9.1.	Spis rysunków.....	56
9.2.	Spis tabel.....	57
9.3.	Spis listingów.....	58

2. Wstęp

Telefony komórkowe po raz pierwszy zagościły w świadomości ludzi w roku 1973, gdy Motorola zaprezentowała pierwszy prototyp przenośnego telefonu [1]. Wtedy też, trzeciego kwietnia, została wykonana pierwsza rozmowa z użyciem tego urządzenia, kiedy to pracownik dokonujący prezentacji postanowił zatelefonować do konkurencyjnej firmy telekomunikacyjnej. Urządzenie zostało jednak wypuszczone na rynek dopiero 10 lat później w roku 1983 pod nazwą DynaTAC 8000X. Oferowało ono wtedy 6 godzin pracy w stanie gotowości, lub 30 minut rozmowy, przy jednoczesnej wadze niemal kilograma. Ówczesnie główną grupą docelową konsumentów tego produktu byli majątni biznesmeni, którym nie straszna była cena w wysokości około czterech tysięcy dolarów. Zapewne nikt wtedy nie podejrzewał, że już 20 lat później ponad 60% Amerykanów będzie w posiadaniu telefonu komórkowego, który mieścił się w kieszeni [2]. W roku 2021 było to już 97%, przy czym 83% użytkowników korzystało z inteligentnego urządzenia typu smartphone. Statystyki bardzo podobnie malują się w skali świata, gdzie szacuje się ponad 6,5 miliarda aktywnych użytkowników. Stanowi około 83% populacji [3]. Rysunek 2.1 przedstawia wykres liczby użytkowników smartphone'ów na świecie od roku 2016, wraz z prognozami na przyszłe lata.



Rysunek 2.1 Wykres liczby użytkowników telefonów typu smartphone na świecie w latach 2016-2026 (prognozowane). Źródło: [3]

Smartfony, poza pełnieniem funkcji komunikacyjnej zapewniają w tym momencie tyle różnych możliwości i opcji, że już od dawna zasługują na miano „kieszonkowych komputerów”. Zaczynając od dostarczania swoim użytkownikom rozrywki aż po zastąpienie pewnych narzędzi (np. aparatów fotograficznych). Mogą one również pełnić funkcje podręcznych terminarzy i notatników.

Ludzie posiadają różne nawyki w kwestii opieki nad roślinami domowymi. Niektórzy nadają im imiona niemalże je personalizując i codziennie je doglądając, a inni ze względu na szybki tryb życia, lub zdiagnozowane zaburzenia uwagi potrzebują pomocy w pamiętaniu o zabiegach pielęgnacyjnych. Szczególnie przy posiadaniu dużej liczby roślin wielu gatunków pamiętanie o zróżnicowanych potrzebach ich wszystkich może sprawiać problemy.

Jednocześnie, podczas trwania pandemii Covid-19 zapotrzebowanie na rośliny domowe wzrosło aż o 20% [4] i dalej rośnie. Razem z tym zwiększyło się również zainteresowanie odmianami kolekcjonerskimi [5], które, w dobie kiedy inne dobra luksusowe nie mają zastosowania ze względu na ograniczenia w opuszczaniu domu, stanowią nowy obiekt westchnień wielu osób. Tym samym, ludzie, zamiast inwestować w drogie zegarki i torebki, inwestują w rośliny, których ceny nierzadko osiągają wiele tysięcy złotych. Ci, których nie stać na odmiany kolekcjonerskie, zadawalają się kolekcjonowaniem tych łatwiej dostępnych i tańszych.

W związku z tym pojawił się pomysł zrealizowania aplikacji, która, nie dość, że pomagałaby użytkownikom opiekować się roślinami, oferując szeroki wachlarz rodzajów przypomnień, to dodatkowo pomagałaby im w przechowywaniu ich aktualnej listy upragnionych roślin tak aby była zawsze pod ręką, kiedy posiadają wolną gotówkę i próbują sobie przypomnieć jakimi odmianami byli zainteresowani. Dodatkowo mogłaby ona uczyć dobrych nawyków pielęgnacyjnych, jak na przykład sprawdzanie przed podlaniem roślin czy na pewno ziemia wystarczająco przeschła.

Aplikacja, która zostanie zaprojektowana i zaimplementowana w ramach tej pracy będzie spełniała każdy z powyższych warunków, aby zapewnić użytkownikowi pomoc w zarządzaniu jego domową kolekcją roślin.

3. Cel i zakres pracy

Celem pracy jest zaprojektowanie i implementacja aplikacji mobilnej na platformę Android ułatwiającej opiekę nad roślinami doniczkowymi.

W ramach celu ogólnego wyznaczone zostały następujące cele cząstkowe:

- analiza istniejących rozwiązań
- opis technologii wykorzystanych podczas implementacji
- opracowanie projektu aplikacji
- implementacja

3.1. Analiza istniejących rozwiązań

Realizacja tego celu opierać się będzie na przestudiowaniu istniejących rozwiązań w tym aplikacji Waterbot, Planteo, Garden Manager, Plant Care Reminder, Vera czy inteligentnej doniczki Xiaomi RoPot oraz wyciągnięcie wniosków z ich funkcjonalności i popularności wśród użytkowników.

3.2. Opis technologii

W ramach tego celu opisane zostaną następujące technologie: Android OS; Android Studio IDE (ang. Integrated Development Environment – zintegrowane środowisko programistyczne); VCS (ang. Version Control System – system kontroli wersji) Git; język programowania Java; język programowania Kotlin; LiveData; ViewModel; Room; Koin.

3.3. Projekt Aplikacji

Wykonany zostanie projekt aplikacji zawierający: opis wymagań funkcjonalnych i нефункциональных, diagram przypadków użycia, scenariusze przypadków użycia, schemat bazy danych, diagram klas oraz projekty ekranów interfejsu użytkownika aplikacji.

3.4. Implementacja

Zrealizowana zostanie implementacja aplikacji, następnie wykonany opis środowiska, modułów aplikacji, wzorców projektowych wykorzystanych podczas realizacji, problemów napotkanych w drodze procesu implementacji oraz testów jednostkowych, integracyjnych oraz manualnych.

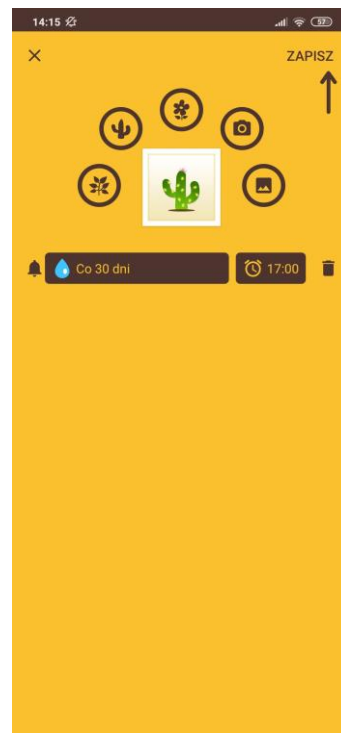
4. Analiza istniejących rozwiązań

4.1. Wstęp

Poniższa analiza powstała w celu określenia zalet i wad istniejących na rynku rozwiązań, mających wspomagać opiekę nad roślinami doniczkowymi. Wszystkie wymienione rozwiązania są dostępne na platformę Android.

4.1. Waterbot

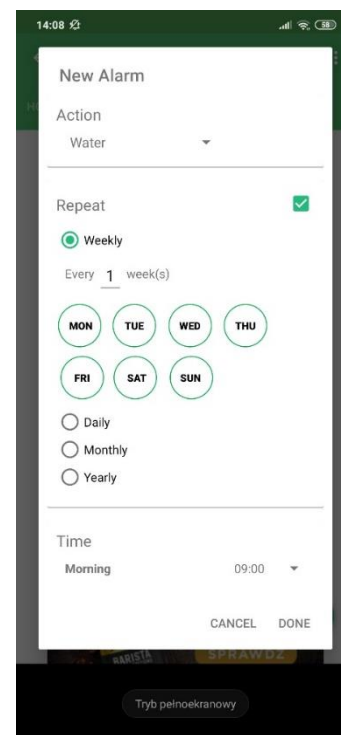
Już na pierwszy rzut oka aplikacja cechuje się nietypową kolorystyką dla tej tematyki (połączenie żółci i brązu) oraz subiektywnie mało intuicyjnym interfejsem użytkownika. Podczas dodawania nowej rośliny dostępne opcje ustawienia grafiki reprezentowane są przy pomocy ikon, które za wyjątkiem ikony aparatu i galerii przedstawiają trzy różne kategorie. Wybór takiego przedstawienia opcji edycji identyfikatora wydaje mi się nieintuicyjny względem innych obecnych na rynku rozwiązań z różnych dziedzin. Nadanie roślinie nazwy, lub utworzenie powiadomienia dotyczącego nawożenia, czy notatki, możemy wykonać dopiero przy edycji elementu. Wydaje się to potencjalnie uciążliwe, jeżeli nowy użytkownik dodaje ich wiele. Aplikacja teoretycznie oferuje dodatkowe ustawienia, jednak, po wybraniu tej opcji, jedyne co zostaje zaprezentowane, to promocja innych produktów tego samego wytwórcy, możliwość oceny, polityka prywatności i zasady użytkowania. Mimo wszystko narzędzie Waterbot w sklepie play cieszy się ponad sto tysięcy pobrań oraz pięćdziesiąt tysięcy ocen, ma również najwyższą ocenę spośród omawianych przez mnie rozwiązań – 4,4. Rysunek 4.1 przedstawia interfejs dodawania nowej rośliny omawianej aplikacji.



Rysunek 4.1: Aplikacja Waterbot - dodawanie nowej rośliny. Źródło: [35]

4.2. Planteo

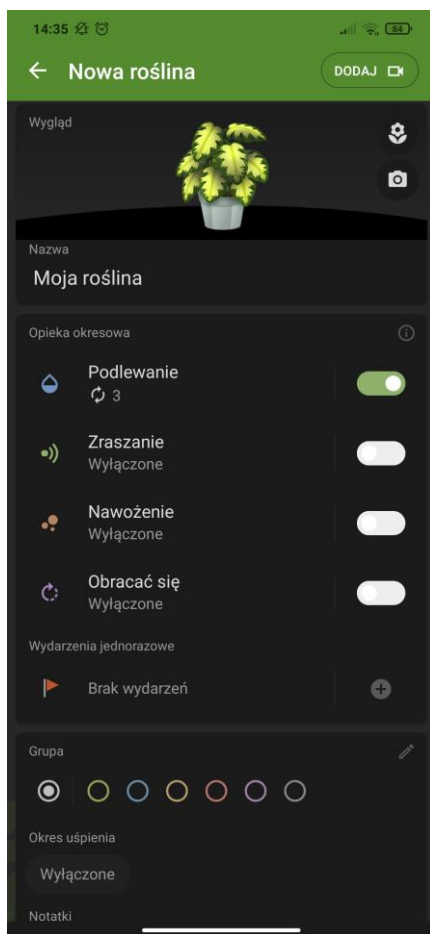
Planteo to kolejna aplikacja do wspomagania opieki nad roślinami domowymi, analizie poddaje jej darmową wersję. W sklepie play cieszy się zdecydowanie mniejszą popularnością niż poprzednia (ponad 5 tysięcy pobrań, 125 ocen) oraz najniższą oceną spośród omawianych – 3,9. Mimo to wydaje mi się znacznie bardziej przyjazna wizualnie oraz oferuje pewne dodatkowe funkcje w stosunku do Waterbot – 12 różnych opcji tematycznych alarmów, personalizowalne powtarzanie alarmów – z nawet roczną częstotliwością. Wartość dodaną stanowi również dziennik, wpisy w którym mogą się odnosić do tych samych dwunastu akcji co alarmy. Istnieje również możliwość dodania wielu zdjęć każdej rośliny, co potencjalnie pozwala widzieć ich zmiany w czasie. Rysunek 4.2 przedstawia ekran dodawania nowego powiadomienia dla rośliny.



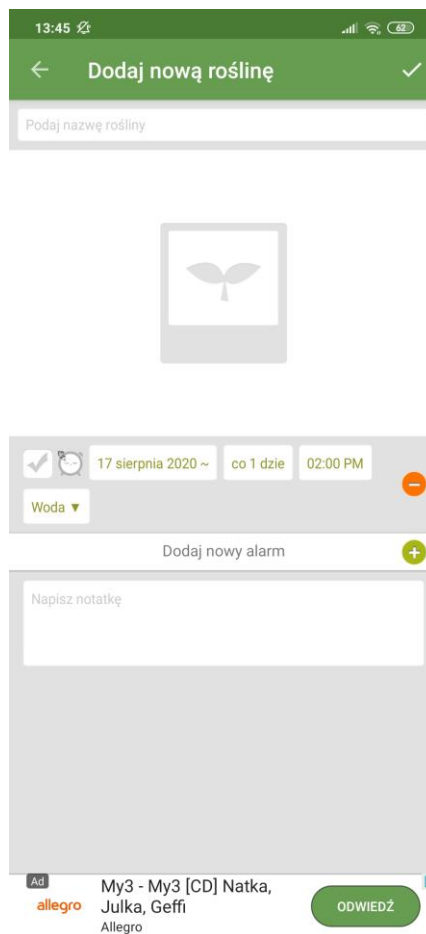
Rysunek 4.2: Aplikacja Planteo - ustawienia powiadomień dla rośliny. Źródło: [33]

4.3. Garden Manager

W tym przypadku znowu pod lupą znalazła się jedna z bardziej popularnych aplikacji z ponad stoma tysiącami pobrań i ponad tysiącem ocen, które uśredniły się do 4,0. Estetyka aplikacji wydaje się na podobnym poziomie co Planteo, na zdecydowany plus przemawia jednak fakt istnienia polskiej wersji językowej. Poza nią niestety nie znajdziemy tutaj interesujących dodatkowych funkcji względem poprzedniczek. Niewykorzystany potencjał wydaje się leżeć w jednej, prawdopodobnie nie najlepiej przetłumaczonej – „przeszukaj najbliższy sklep ogrodniczy” – która to kieruje użytkownika do Map Google z wynikiem wyszukiwania „sklep ogrodniczy”. Rysunek 4.4 przedstawia ekran aplikacji dotyczący dodawania nowej rośliny.



Rysunek 4.3: Aplikacja Plant Care Reminder - dodawanie nowej rośliny
Źródło: [36]



Rysunek 4.4: Aplikacja Garden Manager - dodawanie nowej rośliny
Źródło: [32]

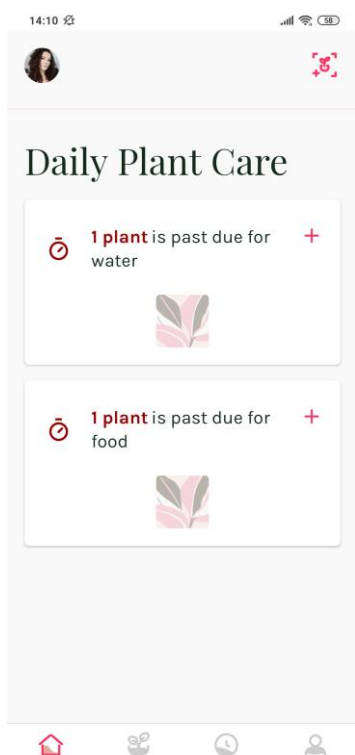
4.4. Plant Care Reminder

Kolejna aplikacja cieszy się podobną popularnością co poprzednia, jednocześnie uzyskując nieco wyższą średnią ocenę – 4,2. Interfejs wydaje się być nieco bardziej dopracowany estetycznie. Pod tym względem należy również zwrócić uwagę na obecność trybu ciemnego dostosowującego się do trybu urządzenia, co nie było obecne w żadnym z poprzednich rozwiązań. Ciekawym akcentem wydaje się również graficzna wizualizacja poziomu nawodnienia rośliny korespondująca z liczbą dni pozostałych do kolejnego podlewania. Niestety aplikacja dla każdej oferuje jedynie trzy dodatkowe opcje pielęgnacyjne – zraszanie, nawożenie i prawdopodobnie źle przetłumaczone na potrzeby polskiej wersji językowej „obracać się”. Podczas dodawania lub edycji rośliny istnieje możliwość określenia

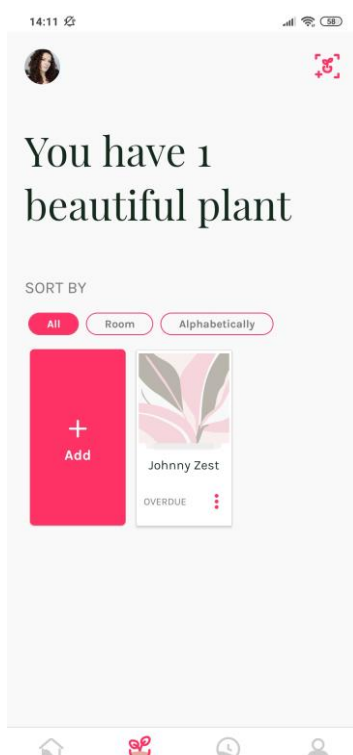
dla niej „grupy”, których zdecydowanym minusem jest stała, ograniczona liczba. Dodatkowo ich jedyną funkcją jest to, że stanowią potencjalny aspekt do posortowania listy. Nie można niestety wykonać jakiejś akcji dla „grupy”, jedynie dla wszystkich roślin, albo dla tych, które według aplikacji wymagają opieki. Niepotrzebnym i niecodziennym dodatkiem w ramach tej tematyki wydają mi się dwie jednodominujące gry z maskotkami aplikacji.

4.5. Vera

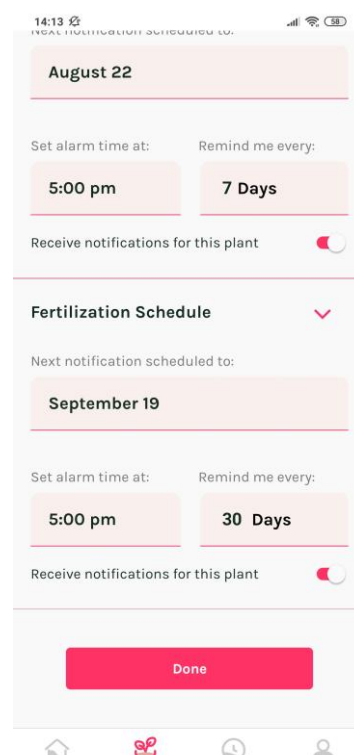
Ostatnie omawiane przeze mnie mobilne rozwiązanie uzyskało tę samą średnią ocenę i podobną popularność co poprzedniczka. Na tle innych wyróżnia się bardzo dopracowanym i estetycznym interfejsem użytkownika. Dodawanie nowej rośliny jest rozdzielone na kilka etapów. Użytkownik ma również wybór, czy chce od razu ustalić harmonogram podlewania lub nawożenia, czy może woli zrobić to później. Pewnym minusem jednak jest bardzo okrojony wybór zabiegów pielęgnacyjnych – dostępne są tylko dwa wymienione powyżej. Ciekawym rozwiązaniem za to jest podział aplikacji na cztery ekrany. Na pierwszym z nich pojawiają się powiadomienia dotyczące ustalonych harmonogramów, na drugim zebrane są wszystkie posiadane rośliny. Kolejny dotyczy podlewania – użytkownik może zaznaczyć pokój spośród samodzielnie dodanych i oznaczyć jako podlany. Być może funkcjonalność ta mogłaby zostać rozszerzona o dostępną w aplikacji akcję nawożenia. Ostatni panel zawiera odnośniki do polityki prywatności, FAQ, oceny, kontaktu i testu systemu powiadomień. Rysunki 4.6-4.8 przedstawiają wybrane ekrany aplikacji.



Rysunek 4.6: Aplikacja Vera - lista zadań do wykonania. Źródło: [34]



Rysunek 1.7: Aplikacja Vera - lista roślin. Źródło: [34]



Rysunek 4.8: Aplikacja Vera - dodawanie rośliny, harmonogram. Źródło: [34]

4.6. Smart doniczka – Xiaomi RoPot

Oprócz rozwiązań w postaci samodzielnych aplikacji mobilnych na rynku pojawiły się również tak zwane smart doniczki. Ta omawiana przeze mnie łączy się z urządzeniem Android za pomocą Bluetooth oraz posiada dedykowaną aplikację do obsługi, za pomocą której możemy mieć dostęp do wartości monitorowanych parametrów. Jako obiektywne plusy rozwiązania uznać można dokładne monitorowanie przez doniczkę poziomów nawilżenia oraz żyzności gleby. Komunikuje się ona z użytkownikiem za pomocą diody, która świeci się na czerwono w przypadku kiedy któryś z parametrów wykracza poza bezpieczne dla rośliny granice. Niestety aplikacja sama w sobie nie oferuje powiadomień, przez co może być trudniej zauważyć potrzeby rośliny jeżeli posiadamy ich więcej. Kolejnym minusem jest konieczność ładowania doniczki. Całe szczęście jedno wystarcza na około 60 dni działania urządzenia, jednak mimo wszystko wydaje mi się, że łatwo można zapomnieć o tym kiedy miało miejsce ostatnie, a rozładowana – staje się zwykłą, drogą doniczką. Kolejny minus – cena. Pojedyncze urządzenie to koszt około 200zł w zależności od miejsca zakupu. Być może jest to akceptowalne jeżeli posiada się jedną roślinę, jednak na dłuższą metę wydaje się nieekonomiczne, zwłaszcza, że – jak każde urządzenie elektroniczne – doniczka może się zepsuć, co nie czyni z niej zakupu tak długotrwałego jak klasyczne donice. Generalizując idea smart doniczki wydaje się być skierowana do osób, które posiadają jedną lub dwie rośliny i nie są zainteresowane zdobywaniem wiedzy na ich temat, co kompletnie rozmija się z moim docelowym użytkownikiem przez co nie jest porównywalne w tych samych kategoriach.

4.7. Podsumowanie

Przeanalizowane rozwiązania w większości posiadają interesujące cechy, jednak moim zdaniem żadne z nich nie zaspokaja wszystkich potrzeb opisanych przeze mnie we wstępie pracy. Ciekawym punktem potencjalnego rozwoju niniejszej aplikacji jest coś, czego na obecną chwilę nie przewiduję w projekcie – czyli możliwość dodawania zdjęć (jednego lub wielu) rośliny przez użytkownika. Dodatkowo, żadne z nich nie oferuje możliwości przechowywania listy życzeń użytkownika. Skupiają się one tylko i wyłącznie na opiece nad roślinami, niekoniecznie dobrze realizując nawet to zadanie.

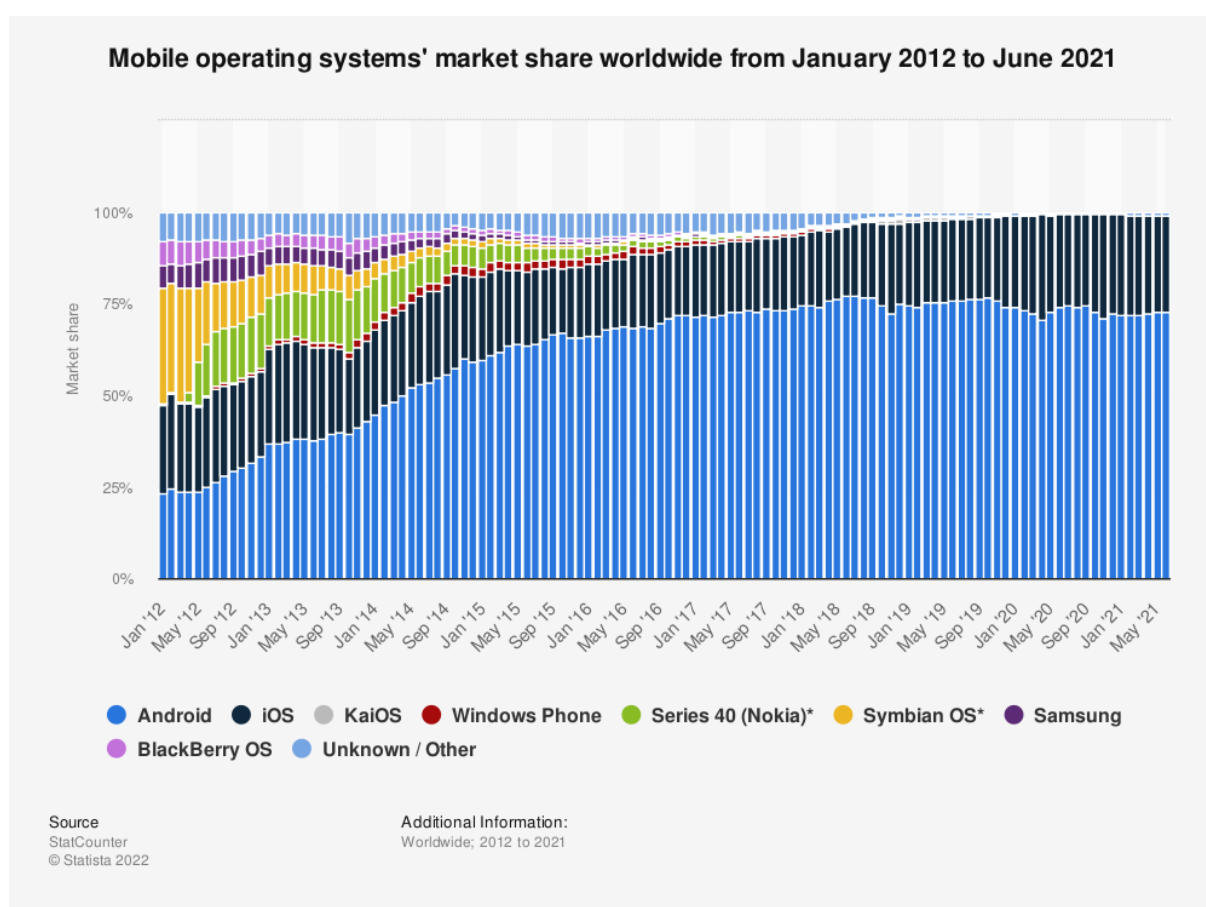
5. Wykorzystane technologie

5.1. Wstęp

W poniższym rozdziale opisane zostaną technologie wykorzystane podczas projektowania, a następnie implementacji aplikacji Planter.

5.2. System Android

Realizowana aplikacja będzie działać w oparciu o system operacyjny Android. Android ma swoje początki w roku 2003 w firmie pod tą samą nazwą. Został on stworzony z myślą o „inteligentniejszych urządzeniach mobilnych, bardziej świadomych lokalizacji i preferencji swojego właściciela”[6]. Już dwa lata później firma została kupiona przez przedsiębiorstwo Google. Pierwsza wersja Androida 1.0 została udostępniona deweloperom w 2007 roku, jego debiut komercyjny miał miejsce rok później, zaś mniej więcej od roku 2010 posiada on większość rynkową.



Rysunek 5.1: Wykres przedstawiający procentowy udział w rynku poszczególnych systemów operacyjnych na urządzenia mobilne w latach 2012-2021. Źródło: [7]

Bardzo ważną z dzisiejszego punktu widzenia decyzją, było użycie systemu Linux jako podstawy dla Androida, ponieważ daje to możliwość oferowania go za darmo osobom trzecim. Z tego względu mniejsi producenci inteligentnych urządzeń przenośnych decydują się na modyfikację, lub wprost użycie dostępnego oprogramowania w swoich produktach, co zdecydowanie wpływa na jego popularyzację. Darmowość samego systemu nie oznacza jednak darmowości oprogramowania wytworzonego na jego podstawie. Kod źródłowy napisany przez programistę stanowi własność intelektualną jego samego, lub firmy, w której pracuje, a gotowe aplikacje są najczęściej dostępne do pobrania w sklepie Play, za darmo lub za opłatą.

Tworzenie oprogramowania na system Android wiąże się z koniecznością dopasowywania się przez programistę do stosunkowo często publikowanych nowych wersji systemu. Oferują one usprawnienia i do pewnego stopnia są ze sobą zgodne, jednak tworzenie oprogramowania, które obsługiwałoby wszystkie wersje byłoby nieefektywne, ponieważ z biegiem czasu zmienia się udział rynkowy poszczególnych z nich.

*Tabela 5.1: Udział rynkowy poszczególnych wersji systemu Android.
Stan – Styczeń 2022. Źródło: [25]*

Wersja	Nazwa	API	Udział rynkowy
5.1	Lollipop	22	1,97%
6	Marshmallow	23	3,31%
7.0	Nougat	24	3,2%
7.1	Nougat	25	1,96%
8.0	Oreo	26	3,12%
8.1	Oreo	27	7,06%
9.0	Pie	28	13,46%
10.0	Queen Cake	29	27%
11.0	Red Velvet Cake	30	35,37%
-	Pozostałe	-	3,55%

Z tego względu z reguły w projekcie określa się trzy różne stałe definiujące zakres wersji, z którymi będzie zgodny produkt:

- `minSdkVersion` – minimalne SDK, z którym będzie zachowana zgodność aplikacji (w aplikacji Planter – 24, Android 7.0 Nougat)
- `compileSdkVersion` – wersja SDK używana podczas kompilacji (w aplikacji Planter – 30, Android 11.0 Red Velvet Cake)
- `targetSdkVersion` – wersja docelowa; uznaje się, że oznacza, że aplikacja została przetestowana na wszystkich wersjach pomiędzy i włącznie z nią i `minSdkVersion` (w aplikacji Planter – 30, Android 11.0 Red Velvet Cake)

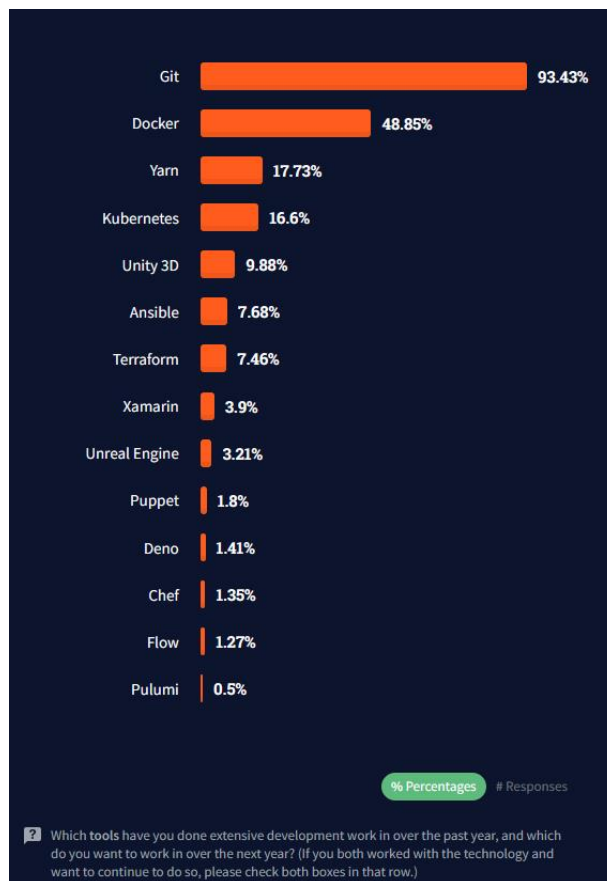
Dzięki powyższym ustawieniom aplikacja będzie możliwa do zainstalowania na około 91% urządzeń (Tabela 5.1), a także zapewnione będą wystarczająco aktualne funkcje systemu (Android 7.0 został wypuszczony na rynek w roku 2016) [8].

5.3. Git

Git jest to system kontroli wersji (VCS – ang. Version Control System) mający swoje początki w 2005 roku. Był on wtedy odpowiedzią na nagłe zapotrzebowanie na nowy VCS wśród społeczności pracującej nad jądrem Linux. Szesnaście lat później Git jest zdecydowanym liderem w swojej dziedzinie. Korzystanie z niego stanowi niemal jedną z podstawowych umiejętności programisty, ze względu na jego powszechne użycie w przedsiębiorstwach wytwarzających różnego typu oprogramowanie. Wśród użytkowników Gita znajdują się nawet tak olbrzymie firmy jak Google czy Microsoft. Najlepszym wyznacznikiem skali dominacji rynkowej systemu Git są jednak wyniki dorocznej ankiety użytkowników popularnego serwisu Stack Overflow (oferującego możliwość zadawania pytań z szeroko pojętego zakresu wytwarzania oprogramowania, oraz udzielania odpowiedzi innym użytkownikom), według których ponad 93% ankietowanych regularnie korzysta z systemu kontroli wersji Git (Rysunek 5.2).

Wyróżnikami Gita, które być może zapewniły mu tak powszechną dominację, są: fakt, że jest on rozproszonym systemem kontroli wersji oraz możliwość pracy offline, bez stałego połączenia ze zdalnym repozytorium. W modelu rozproszonym każdy kto pracuje nad danym fragmentem kodu może to robić na osobnej kopii tego samego pliku. Git przechowuje historię zmian wszystkich plików we wszystkich lokalizacjach tym samym zapewniając sprawniejsze łączenie zmian. Każda taka paczka zmian (ang. commit) stanowi część większej gałęzi zmian (ang. branch) [10]. W każdej chwili gałęzi można ze sobą złączyć za pomocą operacji merge, lub też utworzyć nową gałąź projektu.

Wszystko to umożliwia sprawne przełączanie się między różnymi wersjami tego samego projektu, równoległą pracę nad różnymi pomniejszymi jego fragmentami, czy też łatwiejsze lokalizowanie i naprawianie błędów.



Rysunek 5.2: Wykres ilustrujący popularność narzędzia Git wśród użytkowników portalu StackOverflow. Źródło: [9]

W niniejszym projekcie poza lokalnym korzystaniem z Gita głównie za pomocą graficznego interfejsu zintegrowanego z Android Studio (ale również częściowo za pomocą terminalu CYGWIN zapewniającego interfejs tekstowy), kod aplikacji będzie przechowywany w prywatnym zdalnym repozytorium na portalu GitHub. Zapewnia to ochronę przed utratą kodu, jak również umożliwia pracę z dowolnego urządzenia w dowolnym czasie.

5.4. Java

Java jest językiem programowania wypuszczonym na rynek w roku 1995. Od wielu lat stale plasuje się ona w czołówce najszerzej wykorzystywanych języków programowania oraz jest regularnie aktualizowana.

Nie bez wpływu na jej popularność jest jej prostota – w implementacji Javy pominiętych została pewna ilość funkcji obecnych w języku C/C++, które są rzadko używane lub łączone z nienajlepszymi praktykami kodowania. Dodatkowo jest to język bardzo bezpieczny ze względu na wyeliminowanie możliwych do wystąpienia w jej poprzedniku błędów i problemów z pamięcią. Co jednak najistotniejsze wspiera ona również operowanie na klasach, obiektowość i współbieżność [11].

System Android sam w sobie opiera się przede wszystkim na zmodyfikowanym jądrze Linuksowym oraz na maszynie wirtualnej Javy [12]. Również Android API, które stanowi interfejs programisty aplikacji zostało napisane w tym języku, tym samym na stałe wiążąc go i czyniąc go integralną częścią systemu i narzędzi jemu dedykowanych.

5.5. Kotlin

Kotlin został wypuszczony na rynek w roku 2016 jako język programowania mający stanowić usprawnienie Javy. Został on wyprodukowany przez firmę JetBrains i jest dostępny do użytku pod licencją open-source (Apache 2.0). Pozwala on zarówno na pisanie kodu obiektowo jak i funkcjonalnie oraz umożliwia mieszanie tych dwóch stylów na przestrzeni tego samego projektu. Jego głównym usprawnieniem względem Javy jest znaczna redukcja liczby linii kodu, sięgająca średnio 40% [13]. Mimo, że różnią się składnią, to zapewniona jest możliwość sprawnej interakcji między kodem napisanym w obu wspomnianych językach.

Kotlin jest szeroko wykorzystywany w implementacji aplikacji na platformę Android jednak nie jest to narzędzie dedykowane wyłącznie temu. Jest to język bardzo elastyczny i podobnie jak Java pozwala również na realizację projektów webowych, czy projektów z zakresu data science. Dodatkowo wiele bibliotek dedykowanych wcześniej Javie ma obecnie swoje odpowiedniki w Kotlinie, tym samym coraz bardziej zwiększając jego przewagę nad poprzednikiem.

Podsumowując, najistotniejszymi różnicami pomiędzy dwoma wymienionymi językami, są[14]:

- Kotlin domyślnie nie pozwala na przypisywanie do obiektów wartości null, tym samym zwiększając odporność kodu na wyjątek NullPointerException. W razie potrzeby programista może to jednak łatwo ominąć, poprzez użycie znaków zapytania przy deklaracji zmiennej.
- Kotlin pozwala na rozszerzanie funkcjonalności klas bez konieczności dziedziczenia za pomocą tzw. metod rozszerzających (ang. Extension functions).
- Kotlin zapewnia wsparcie współbieżności wątków z jednoczesną redukcją ich liczby.
- Innym ciekawym dodatkiem jest słowo kluczowe data pozwalające prosto określić klasę jako przechowującą dane. Przy jego użyciu kompilator automatycznie buduje konstruktor klasy oraz funkcje getterów i setterów pól.
- Kotlin oferuje możliwość zarówno programowania funkcjonalnego jak i obiektowego.
- Kotlin nie wspiera publicznych pól klasy, tym samym zwiększając bezpieczeństwo klas (jeżeli programista chce korzystać z pól klasy na zewnątrz niej, musi on samodzielnie zapewnić funkcje getterów i setterów).

W kontekście aplikacji Planter, Kotlin będzie stanowił główny język programowania wykorzystany przy implementacji. Od roku 2017 jest on wspierany przez Android Studio IDE, a zmniejszenie liczby linii kodu powinno przyspieszyć i usprawnić pracę.

5.6. Material Design

Material Design to zestaw reguł i zasobów projektowych zaproponowany przez Google aby wspomóc deweloperów w budowaniu wysokiej jakości interfejsów użytkownika. Jego twórcy czerpią inspirację z druku, gdzie rozmiar, kolor i rozmieszczenie poszczególnych elementów interfejsu nadaje im znaczenie i prowadzi użytkownika zanim jeszcze tekst i ilustracje zostaną przez niego poddane interpretacji. Material Design zapewnia gotowe, interaktywne i do pewnego stopnia personalizowalne komponenty UI, które pozwalają zachować wizualną jednolitość produkowanych aplikacji. Są one dodatkowo skalowalne, co promuje skalowalność wytworzonych z ich użyciem produktów.

5.7. Firebase

Firebase jest platformą spod ramienia Google mającą za zadanie usprawniać budowę i rozwój różnej maści aplikacji wytwarzanych przez programistów na całym świecie. Oferuje ona między innymi narzędzia analityczne, statystyki, wsparcie autentykacji użytkowników, bazy danych czasu rzeczywistego, czy miejsce do przechowywania plików[16]. Oprogramowanie klienckie Firebase pozwala również na bezpośrednią interakcję z jej produktami z poziomu wytwarzanej aplikacji.

W kontekście aplikacji Planter będę korzystać z autentykacji użytkowników oferowanej przez Firebase oraz zewnętrznej bazy danych czasu rzeczywistego do przechowywania ich danych. Wyjątkową cechą autentykacji Firebase jest łatwość i pewność zapewnienia bezpieczeństwa logowania, co jest elementem potrafiącym sprawiać duże problemy przy samodzielnej implementacji.

5.8. Android Studio

Android Studio jest to oficjalne IDE (ang. Integrated Development Environment – zintegrowane środowisko wytwarzania oprogramowania) dedykowane wytwarzaniu aplikacji działających w systemie Android [17]. Zostało ono wytworzone przez przedsiębiorstwo Google we współpracy z firmą JetBrains, która to dostarcza środowisk dla różnych platform i języków programowania. Narzędzie Android Studio jest rozwinięciem istniejącego wcześniej IntelliJ IDEA, służącego do wytwarzania oprogramowania w języku Java. Zostało ono rozszerzone o funkcjonalności i komponenty, mające za zadanie usprawnić i ułatwić wytwarzanie aplikacji mobilnych, takie jak:

- System budowy aplikacji w oparciu o Gradle,
- Szybki i sprawny emulator urządzeń oferujący wiele opcji deweloperskich,
- Możliwość integracji z systemem kontroli wersji,
- Ogromną ilość narzędzi i pluginów tworzonych przez społeczność,
- Narzędzia i frameworki przeznaczone do testowania oprogramowania,
- Narzędzia debugowania,
- Wbudowane wsparcie platformy Google Cloud.

5.9. Android Architecture Components

Android Architecture Components to zbiór bibliotek wypuszczony na rynek przez firmę Google w Listopadzie 2017 roku, mający za zadanie ułatwienie implementacji aplikacji działających w systemie Android poprzez oferowane gotowe rozwiązania popularnych problemów implementacyjnych. Oferowane klasy odnoszą się do architektury aplikacji, utrzymywania trwałości i aktualności przechowywanych i przetwarzanych danych oraz do zarządzania cyklem życia elementów składowych interfejsu użytkownika [18].

5.9.1. Live Data

Live Data to biblioteka, która pozwala elementom interfejsu aplikacji na monitorowanie zmian modelu i reagowanie w odpowiedni sposób. Live Data ma również świadomość cyklu życia komponentów interfejsu i żąda od nich zmian tylko wtedy, kiedy są one w odpowiednim momencie cyklu [19]. Wspomniana biblioteka do funkcjonowania potrzebuje obiektu obserwatora (ang. Observer). Jego stan (aktywny, nieaktywny) mówi o tym, czy dany element interfejsu jest w tym momencie widoczny, a komponent jest informowany o zmianach tylko w momencie kiedy obserwator jest aktywny.

Korzystając z tej biblioteki można być zatem pewnym, że interfejs użytkownika odpowiednio odzwierciedla aktualny stan lokalnej bazy danych. Dodatkowo gwarantuje ona automatyczne wyrejestrowywanie obiektów nasłuchujących na zmiany.

5.9.2. ViewModel

ViewModel to klasa, która została zaprojektowana w celu przechowywania i przetwarzania danych powiązanych z interfejsem użytkownika. Pozwala on danym na przetrwanie zmian interfejsu, takich jak te wynikające ze zmian orientacji ekranu urządzenia [20].

Ze względu na sposób w jaki framework Android zarządza cyklem życia kontrolerów interfejsu użytkownika, obsługa takich zmianami bez użycia biblioteki marnuje bardzo dużo zasobów, ponieważ implikuje konieczność powtarzania pewnych akcji w celu wygenerowania z powrotem utraconych danych. Uzależnienie przechowywania danych od klas obsługujących interfejs użytkownika również nie jest dobrym rozwiązaniem, ponieważ przenosi to na nie zbyt wiele odpowiedzialności i generuje dodatkowy kod, który zmniejsza czytelność plików. Mając to na uwadze istnienie klasy, która odpowiada tylko i wyłącznie za przechowywanie potrzebnych danych jest jak najbardziej uzasadnione i jak najbardziej może być w stanie przyspieszyć działanie aplikacji.

5.9.3. Room

Room to biblioteka skupiająca się na zachowywaniu trwałości danych. Zapewnia ona warstwę abstrakcji nad językiem SQLite i ułatwia mapowanie obiektów względem surowych danych z bazy.

Największe korzyści wynikające z korzystania ze wspomnianej biblioteki są widoczne w aplikacjach, które przechowują lokalnie dużą ilość ustrukturyzowanych danych [21]. W najczęstszym przypadku dane są cache'owane, dzięki czemu są dostępne dla użytkownika nawet w momencie kiedy aplikacja działa offline i nie ma możliwości pobrania danych z serwera.

Room, za pomocą odpowiednich adnotacji klas (@Database, @Entity, @Dao), generuje w tle fragmenty kodu, które w innym przypadku byłyby bardzo powtarzalne i zajmowałyby wiele linii (ang. boilerplate code). Tym samym redukuje jednocześnie czas potrzebny do implementacji aplikacji, jak również czyni kod bardziej czytelnym.

5.9.4. Data Binding

Biblioteka Data Binding pozwala na wiązanie ze sobą elementów interfejsu i zmiennych modelu. Tym samym możliwe jest usprawnienie działania aplikacji poprzez zmniejszenie ilości linii kodu do przetworzenia, czy zapobieganie NullPointerException i wyciekom pamięci [22]. Dzięki temu kod również staje się bardziej czytelny i łatwiejszy w odbiorze.

5.9.5. Navigation

Navigation to komponent mający za zadanie usprawnienie nawigacji na przestrzeni aplikacji [23]. Składa się on z trzech pod-komponentów:

- pliku XML zawierającego graf nawigacyjny, w skład którego wchodzi destynacje (izolowane elementy zawartości, np. fragmenty), jak również możliwe ścieżki którymi użytkownik może poruszać się po aplikacji
- pustego kontenera (NavHost) wyświetlającego destynacje i ścieżki należące do grafu
- obiektu NavController, który zarządza nawigacją aplikacji wewnątrz wspomnianego kontenera.

Podobnie jak w przypadku poprzednio wspomnianych bibliotek Android Architecture Components, jedną z głównych korzyści, poza usprawnieniem działania aplikacji, jest redukcja liczby linii kodu, tym razem poprzez centralizację metod umożliwiających poruszanie się po aplikacji przez użytkownika.

5.10. Koin

Wstrzykiwanie zależności to technika programowania, która pozwala uniezależnić klasę od pozostałych klas. W skali mikro oznacza to np. przekazywanie „gotowych” obiektów do konstruktora klasy, zamiast konstrukcji tego samego obiektu w jej wnętrzu (wstrzykiwanie przez konstruktor). W przypadku nawet nieco większych projektów dobrą praktyką jest korzystanie z pośredniczącej klasy „wstrzykującej”. W celu ułatwienia zarządzania tymi zależnościami powstało zatem wiele bibliotek i frameworków.

Koin jest jednym z nich. Powstał on z myślą o aplikacjach pisanych w języku Kotlin, jako lżejsza i łatwiejsza do opanowania alternatywa dla takich rozwiązań jak np. Dagger2. Wymaga on od programisty zdefiniowania modułów aplikacji wraz z ich wzajemnymi zależnościami[24]. Poza zarządzaniem wstrzykiwaniem Koin może również sprawować pieczę nad obecnymi w projekcie singletonami (klasami, których może istnieć tylko jedna aktywna instancja). Ze względu na to, że korzystanie z frameworka Koin mimo wszystko produkuje pewną ilość kodu boilerplate, szczególnie jeżeli implementowany produkt jest bardzo dużym i skomplikowanym projektem, jest on rozwiązaniem bardziej odpowiednim dla mniejszych projektów.

5.11. Biblioteki zewnętrzne – podsumowanie

Tabela 5.2 zawiera podsumowanie zewnętrznych bibliotek, które zostaną wykorzystane podczas implementacji aplikacji Planter.

Tabela 5.2: Podsumowanie bibliotek zewnętrznych, użytych w trakcie implementacji. Źródło: opracowanie własne.

Lp.	Nazwa	Ścieżka	Zastosowanie
1	Android Architecture Components	androidx.lifecycle	Szerokie zastosowanie w zakresie cyklu życia aplikacji, wliczając takie rozwiązania jak: LiveData, ViewModel, DataBinding.
2	Navigation	androidx.navigation	Biblioteka ułatwiająca zdefiniowanie dróg poruszania się po aplikacji, oraz przełączanie widoków.
3	Room	androidx.room	Biblioteka ułatwiająca implementację bazy danych oraz zapewniająca ich trwałość i przetwarzanie.
4	Koin	org.koin:koin-android:2.2.1	Biblioteka zapewniająca narzędzia do obsługi wstrzykiwania zależności.

6. Zarys projektu

6.1. Wstęp

Poniższy rozdział zawiera zarys projektu aplikacji Planter. Składają się na niego: opis wymagań funkcjonalnych i нефункциональных, przypadki użycia, projekt struktury bazy danych, diagram klas aplikacji oraz projekt interfejsu użytkownika.

6.1. Wymagania

Projekt dowolnego oprogramowania wymaga określenia oczekiwań (z reguły przez klienta), które musi spełniać gotowy produkt. Oczekiwania te formułuje się w postaci wymagań. Zgodnie z literaturą występuje podział na wymagania funkcjonalne i нефункциональные. Wymagania funkcjonalne określają konkretne funkcjonalności, które aplikacja musi posiadać. Wymagania нефункциональные zaś stanowią kryterium do oceny jakości oprogramowania.

6.1.1. Wymagania funkcjonalne

- Wyświetlanie listy posiadanych roślin
- Dodawanie nowej rośliny
- Usuwanie rośliny
- Edycja rośliny
- Wyświetlanie listy roślin pogrupowanych ze względu na pokój
- Wyświetlanie listy przypomnień
- Dodawanie przypomnień
- Usuwanie przypomnień
- Oznaczanie aktywnych przypomnień jako wykonane
- Wyświetlanie kalendarza przypomnień
- Wyświetlanie biblioteki gatunków i odmian roślin
- Dodawanie nowych gatunków i odmian roślin do biblioteki
- Edycja dodanych przez użytkownika gatunków i odmian roślin
- Usuwanie dodanych przez użytkownika gatunków i odmian roślin
- Tworzenie konta użytkownika
- Logowanie
- Zapis kopii danych na serwerze (tylko dla użytkowników zalogowanych)
- Przywracanie danych z kopii (tylko dla użytkowników zalogowanych)
- Dodawanie gatunków/odmian roślin do listy życzeń
- Usuwanie gatunków/odmian roślin z listy życzeń użytkownika

6.1.2. Wymagania нефункциональные

- Wsparcie systemu Android w wersji 7.0 i wyższej
- Wsparcie dla urządzeń mobilnych typu: smartphone, tablet
- Wsparcie dla urządzeń o różnych rozmiarach ekranu
- Obsługa obracania ekranu
- Wykorzystanie Material Design

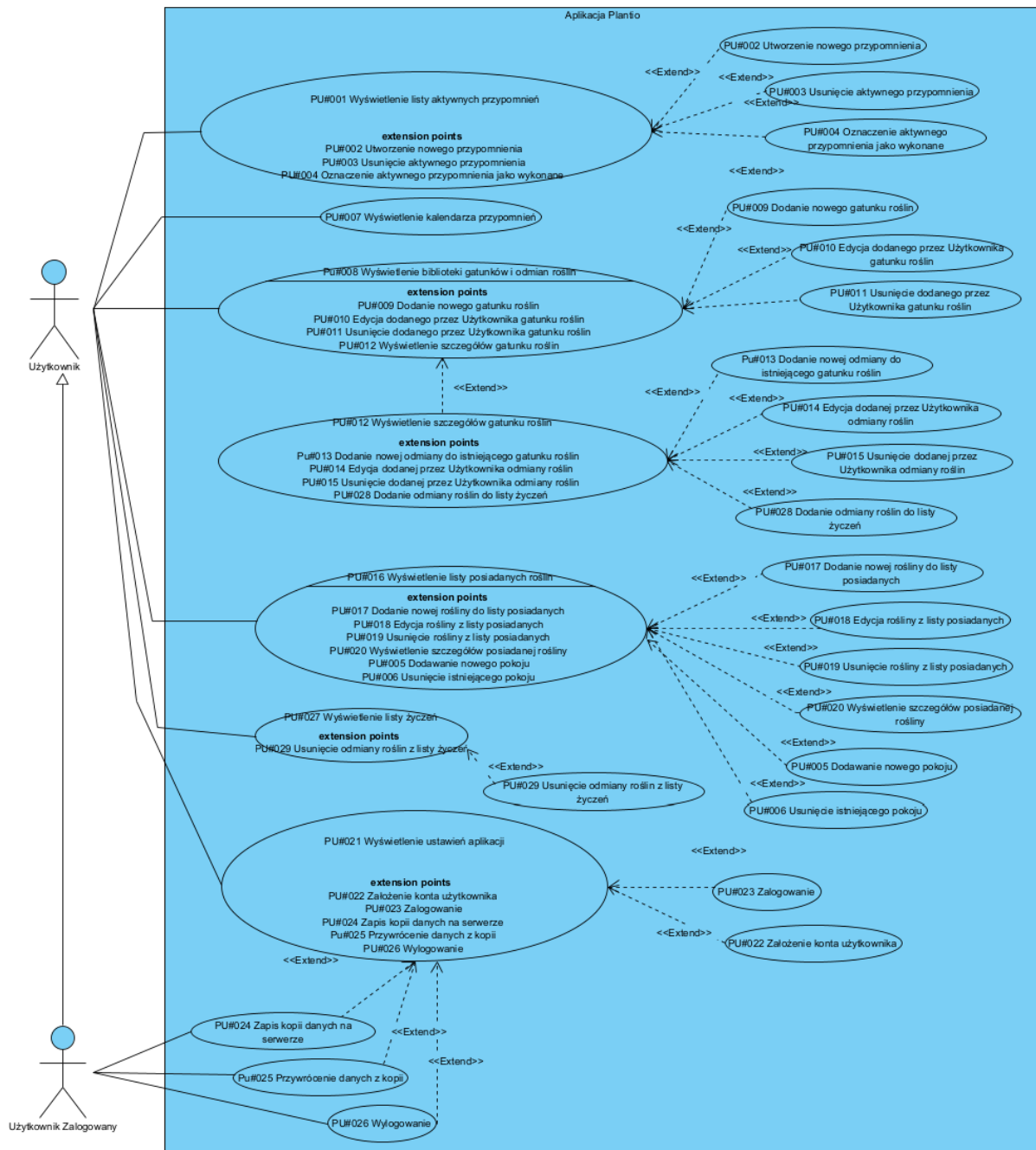
6.2. Przypadki użycia

Kolejnym krokiem po określeniu wymagań dla projektu, jest przekształcenie ich w przypadki użycia. Mają one reprezentować bardziej szczegółowo konkretne akcje, które będą mogły zostać wykonane przez Użytkownika w aplikacji.

- PU#001 Wyświetlenie listy aktywnych przypomnień
- PU#002 Utworzenie nowego przypomnienia
- PU#003 Usunięcie aktywnego przypomnienia
- PU#004 Oznaczenie aktywnego przypomnienia jako wykonane
- PU#005 Dodanie nowego pokoju
- PU#006 Usunięcie istniejącego pokoju
- PU#007 Wyświetlenie kalendarza przypomnień
- PU#008 Wyświetlenie biblioteki gatunków i odmian roślin
- PU#009 Dodanie nowego gatunku roślin
- PU#010 Edycja dodanego przez użytkownika gatunku roślin
- PU#011 Usunięcie dodanego przez użytkownika gatunku roślin
- PU#012 Wyświetlenie szczegółów gatunku roślin
- PU#013 Dodanie nowej odmiany do istniejącego gatunku roślin
- PU#014 Edycja dodanej przez użytkownika odmiany roślin
- PU#015 Usunięcie dodanej przez użytkownika odmiany roślin
- PU#016 Wyświetlenie listy posiadanych roślin
- PU#017 Dodanie nowej rośliny do listy posiadanych
- PU#018 Edycja rośliny z listy posiadanych
- PU#019 Usunięcie rośliny z listy posiadanych
- PU#020 Wyświetlenie szczegółów posiadanej rośliny
- PU#021 Wyświetlenie ustawień aplikacji
- PU#022 Założenie konta Użytkownika
- PU#023 Zalogowanie
- PU#024 Zapis kopii danych na serwerze
- PU#025 Przywrócenie danych z kopii
- PU#026 Wylogowanie
- PU#027 Wyświetlenie listy życzeń
- PU#028 Dodanie odmiany roślin do listy życzeń
- PU#029 Usunięcie odmiany roślin z listy życzeń

6.2.1. Diagram przypadków użycia

Poniższy podrozdział zawiera Rysunek 6.1, który przedstawia diagram wcześniej określonych przypadków użycia.



Rysunek 6.1: Diagram przypadków użycia aplikacji Planter. Źródło: opracowanie własne.

6.2.2. Scenariusze przypadków użycia

Kolejnym istotnym krokiem w drodze wykonywania projektu oprogramowania jest skrupulatne opisanie scenariuszów przypadków użycia, czyli rozszerzenie poprzednich dwóch podrozdziałów o dokładne opisy akcji wykonywanych w ramach każdego przypadku użycia. Scenariusze są przedstawione poniżej w tabelach 6.1-6.29.

Tabela 6.1: PU#001 Wyświetlenie listy aktywnych przypomnień

PU#001	
Nazwa	Wyświetlenie listy aktywnych przypomnień
Aktor	Użytkownik
Opis	Użytkownik chce wyświetlić listę wszystkich aktywnych przypomnień. Domyślnie posortowane są według pokoju i zgodnie z ustawionym czasem.
Warunki początkowe	<ul style="list-style-type: none">• Aplikacja jest zainstalowana
Przebieg główny	<ol style="list-style-type: none">1. Użytkownik włącza aplikację Planty2. Domyślnie zostaje wyświetlona lista aktywnych przypomnień
Przebieg alternatywny	<ol style="list-style-type: none">2a. W przypadku braku aktywnych przypomnień zostaje wyświetlona stosowna informacja.
Warunki końcowe	<ul style="list-style-type: none">• Użytkownik widzi listę aktywnych przypomnień

Tabela 6.2: PU#002 Utworzenie nowego przypomnienia

PU#002	
Nazwa	Utworzenie nowego przypomnienia
Aktor	Użytkownik
Opis	Użytkownik chce utworzyć nowe przypomnienie.
Warunki początkowe	<ul style="list-style-type: none">• Aplikacja jest zainstalowana.• W bazie znajduje się przynajmniej jedna roślina.• Widoczny jest widok listy przypomnień, kalendarza, listy roślin albo widok szczegółów rośliny.
Przebieg główny	<ol style="list-style-type: none">1. Użytkownik wybiera przycisk dodania przypomnienia.2. Wyświetlony zostaje ekran dodawania przypomnienia.3. Użytkownik wybiera rodzaj przypomnienia.4. (Opcjonalne) Użytkownik wybiera, której rośliny dotyczy przypomnienie.5. Użytkownik wybiera czy przypomnienie jest cykliczne czy jednorazowe.6. (Opcjonalne) Użytkownik uzupełnia notatki na temat przypomnienia7. Użytkownik klika na przycisk zapisu przypomnienia.8. Ekran dodawania przypomnienia zostaje zamknięty.
Przebieg alternatywny	<ol style="list-style-type: none">6a. W przypadku gdy został dokonany wybór przypomnienia cyklicznego użytkownik wybiera z długość cyklu. Powrót do kroku 6.8a. W przypadku braku dokonanie wyboru rodzaju przypomnienia wyświetlony zostaje stosowny komunikat, powrót do kroku 3.
Warunki końcowe	<ul style="list-style-type: none">• Nowy rekord przypomnienia zostaje dodany do bazy.

	<ul style="list-style-type: none"> • Jeżeli ostatnim widocznym ekranem był widok kalendarza i w danym dniu nie było ustawione żadne przypomnienie, to widok zostaje odświeżony i dzień jest oznaczony kropką. • Jeżeli ostatnim ekranem był widok listy roślin, to zostaje on odświeżony i roślina, której powiadomienie dotyczy ma zaktualizowany licznik liczby przypomnień cyklicznych lub jednorazowych (w zależności od rodzaju przypomnienia). • Jeżeli ostatnim ekranem był widok rośliny, to zostaje on odświeżony i lista jej przypomnień zawiera nowy rekord.
--	--

Tabela 6.3: PU#003 Usunięcie aktywnego przypomnienia

PU#003	
Nazwa	Usunięcie aktywnego przypomnienia
Aktor	Użytkownik
Opis	Użytkownik chce usunąć przypomnienie
Warunki początkowe	<ul style="list-style-type: none"> • Aplikacja jest zainstalowana • W bazie znajduje się przynajmniej jeden rekord przypomnienia • Widoczna jest lista aktywnych przypomnień albo widok dnia w kalendarzu.
Przebieg główny	<ol style="list-style-type: none"> 1. Użytkownik znajduje interesujący go rekord i klika na niego 2. Wyświetlony zostaje ekran identyczny do ekranu dodawania przypomnienia, jednak wypełniony danymi. 3. Użytkownik klika na przycisk usunięcia przypomnienia. 4. Ekran edycji przypomnienia zostaje zamknięty
Przebieg alternatywny	
Warunki końcowe	<ul style="list-style-type: none"> • Rekord przypomnienia zostaje usunięty z bazy • Jeśli ostatnim widocznym ekranem była lista aktywnych przypomnień to zostaje ona odświeżona i nie zawiera usuniętego rekordu • Jeżeli ostatnim widocznym ekranem był widok kalendarza to zostaje on odświeżony i nie zawiera usuniętego rekordu

Tabela 6.4: PU#004 Oznaczenie przypomnienia jako wykonane

PU#004	
Nazwa	Oznaczenie przypomnienia jako wykonane
Aktor	Użytkownik
Opis	Użytkownik chce oznaczyć aktywne przypomnienie jako wykonane.
Warunki początkowe	<ul style="list-style-type: none"> • Aplikacja jest zainstalowana. • W bazie znajduje się przynajmniej jeden rekord przypomnienia. • Widoczna jest lista aktywnych przypomnień albo widok dnia w kalendarzu.
Przebieg główny	<ol style="list-style-type: none"> 1. Użytkownik znajduje interesujący go rekord i klika na odpowiednią ikonę

	<ol style="list-style-type: none"> Wyświetlony zostaje ekran ze wskazówkami pielęgnacyjnymi związanymi z rodzajem przypomnienia (jeżeli rodzaju przypomnienia dotyczą jakieś wskazówki) oraz z zapytaniem wymagającym potwierdzenia. Użytkownik wybiera odpowiednią ikonę i potwierdza zapytanie. Ekran ze wskazówkami i zapytaniem zostaje zamknięty.
Przebieg alternatywny	3a. Użytkownik wybiera odpowiednią ikonę i anuluje akcję.
Warunki końcowe	<ul style="list-style-type: none"> Jeśli ostatnim widocznym ekranem była lista aktywnych przypomnień to zostaje ona odświeżona i obok wybranego rekordu pojawia się ikona wykonania zadania, a następnie rekord przestaje być wyświetlany. Jeżeli ostatnim widocznym ekranem był widok kalendarza to zostaje on odświeżony i obok wybranego rekordu pojawia się ikona wykonania zadania. Jeżeli ostatnim widocznym ekranem był widok kalendarza i wszystkie zadania w danym dniu zostały wykonane – kropka oznaczająca dany dzień zmienia kolor na zielony.

Tabela 6.5: PU#005 Dodanie nowego pokoju

PU#005	
Nazwa	Dodanie nowego pokoju
Aktor	Użytkownik
Opis	Użytkownik chce dodać nowy pokój.
Warunki początkowe	<ul style="list-style-type: none"> Aplikacja jest zainstalowana. Widoczny jest ekran dodawania/edycji danych rośliny albo ekran listy roślin.
Przebieg główny	<ol style="list-style-type: none"> Użytkownik klika na przycisk dodawania nowego pokoju. Widoczny jest ekran dodawania pokoju. Użytkownik podaje nazwę. Użytkownik zatwierdza klikając przycisk. Ekran zostaje zamknięty.
Przebieg alternatywny	3a. Jeżeli w bazie znajduje się już pokój o takiej samej nazwie wyświetlony zostaje stosowny komunikat. Powrót do punktu 3.
Warunki końcowe	<ul style="list-style-type: none"> Do bazy danych zostaje dodany nowy rekord pokoju. Jeżeli ostatnim widocznym ekranem był ekran dodawania lub edycji rośliny to zostaje on odświeżony i w rozwijanej liście wyboru pokoju widoczny jest nowy rekord. Jeżeli ostatnim widocznym ekranem był ekran listy roślin, to zostaje on odświeżony i widoczny jest nowy rekord w postaci nagłówka pokoju.

Tabela 6.6: PU#006 Usunięcie istniejącego pokoju

PU#006	
Nazwa	Usunięcie istniejącego pokoju
Aktor	Użytkownik

Opis	Użytkownik chce usunąć jeden z istniejących pokoiów.
Warunki początkowe	<ul style="list-style-type: none"> • Aplikacja jest zainstalowana. • Widoczny jest ekran listy roślin. • W bazie znajduje się przynajmniej jeden rekord pokoju.
Przebieg główny	<ol style="list-style-type: none"> 1. Użytkownik znajduje i klika na wybraną pozycję. 2. Rozwinięta zostaje lista dostępnych opcji. 3. Użytkownik znajduje i klika na przycisk usunięcia. 4. Lista zostaje zamknięta. 5. Wyświetlony zostaje ekran potwierdzenia usunięcia pokoju wraz z komunikatem, że rośliny znajdujące się w tym pokoju będą od tego momentu nieprzypisane. 6. Użytkownik potwierdza usunięcie. 7. Ekran zostaje zamknięty.
Przebieg alternatywny	6a. W przypadku braku potwierdzenia ekran zostaje zamknięty i następuje powrót do punktu 1.
Warunki końcowe	<ul style="list-style-type: none"> • Rekord pokoju zostaje usunięty z bazy danych. • Rekordy wszystkich roślin przypisanych wcześniej do pokoju zostają zmienione. • Jeżeli ostatnim widocznym ekranem był widok listy roślin zostaje on odświeżony i nie znajduje się już na nim nagłówek usuniętego pokoju, a przypisane do niego rośliny znajdują się w sekcji ogólnej na górze strony.

Tabela 6.7: PU#007 Wyświetlenie kalendarza przypomnień

PU#007	
Nazwa	Wyświetlenie kalendarza przypomnień
Aktor	Użytkownik
Opis	Użytkownik chce wyświetlić kalendarz przypomnień. Obecność przypomnienia w danym dniu oznaczona jest kropką.
Warunki początkowe	<ul style="list-style-type: none"> • Aplikacja jest zainstalowana
Przebieg główny	<ol style="list-style-type: none"> 1. Użytkownik klika ikonę kalendarza na dole ekranu. 2. Wyświetlony zostaje ekran z widokiem kalendarza, na którym obecność przypomnienia w danym dniu oznaczona jest kropką, a lista obowiązujących w danym dniu przypomnień (domyślnie obecny dzień) znajduje się poniżej kalendarza, na dole ekranu. 3. (Opcjonalne) Użytkownik wybiera interesujący go miesiąc za pomocą strzałek. 4. (Opcjonalne) Użytkownik wybiera interesujący go dzień.
Przebieg alternatywny	
Warunki końcowe	<ul style="list-style-type: none"> • Użytkownik widzi ekran z widokiem kalendarza i dnia

Tabela 6.8: PU#008 Wyświetlenie biblioteki gatunków i odmian roślin

PU#008	
Nazwa	Wyświetlenie biblioteki gatunków i odmian roślin
Aktor	Użytkownik

Opis	Użytkownik chce wyświetlić bibliotekę gatunków roślin.
Warunki początkowe	<ul style="list-style-type: none"> Aplikacja jest zainstalowana.
Przebieg główny	<ol style="list-style-type: none"> Użytkownik klika przycisk biblioteki na dole ekranu. Wyświetlona zostaje posortowana alfabetycznie lista gatunków roślin znajdujących się w bazie.
Przebieg alternatywny	
Warunki końcowe	<ul style="list-style-type: none"> Użytkownik widzi listę gatunków roślin znajdujących się w bazie danych.

Tabela 6.9: PU#009 Dodanie nowego gatunku do biblioteki

PU#009	
Nazwa	Dodanie nowego gatunku do biblioteki
Aktor	Użytkownik
Opis	Użytkownik chce dodać do biblioteki własny gatunek.
Warunki początkowe	<ul style="list-style-type: none"> Aplikacja jest zainstalowana. Widoczny jest ekran biblioteki.
Przebieg główny	<ol style="list-style-type: none"> Użytkownik klika przycisk dodania nowego gatunku roślin. Wyświetlony zostaje ekran dodawania nowego gatunku. Użytkownik wybiera dodawanie gatunku spośród dostępnych opcji (gatunek i odmiana) Użytkownik podaje nazwę gatunku Użytkownik znajduje i wybiera poziom trudności gatunku w pięciostopniowej skali. Użytkownik znajduje i wybiera poziom nasłonecznienia optymalny dla gatunku. Użytkownik znajduje i wybiera najlepsze cechy stanowiska dla gatunku (przynajmniej jedną) Użytkownik znajduje i wybiera wymagania podlewania dla rośliny. (Opcjonalne) Użytkownik wybiera najlepsze cechy gleby dla gatunku. (Opcjonalne) Użytkownik dodaje opis gatunku. (Opcjonalne) Użytkownik dodaje zdjęcie gatunku Użytkownik znajduje i wybiera przycisk zatwierdzenia. Ekran dodawania gatunku zostaje zamknięty.
Przebieg alternatywny	<ol style="list-style-type: none"> 4a. Jeżeli w bibliotece znajduje się gatunek o tej samej nazwie wyświetlony zostaje stosowny komunikat. Powrót do punktu 3. 12a. Jeżeli którekolwiek z wymaganych pól nie zostało wypełnione wyświetlony zostaje stosowny komunikat. Powrót do punktu 12.
Warunki końcowe	<ul style="list-style-type: none"> Nowy rekord gatunku zostaje dodany do bazy danych. Jeżeli ostatnim widocznym ekranem był widok biblioteki, to zostaje on odświeżony i na liście gatunków roślin widnieje nowy rekord.

Tabela 6.10: PU#010 Edycja dodanego przez użytkownika gatunku

PU#010	
Nazwa	Edycja dodanego przez użytkownika gatunku
Aktor	Użytkownik
Opis	Użytkownik chce wprowadzić zmiany w danych własnoręcznie dodanego gatunku
Warunki początkowe	<ul style="list-style-type: none"> • Aplikacja jest zainstalowana. • W bazie znajduje się przynajmniej jeden gatunek roślin dodany przez Użytkownika. • Widoczny jest ekran danych gatunku.
Przebieg główny	<ol style="list-style-type: none"> 1. Użytkownik klika na przycisk edycji. 2. Wyświetlony zostaje ekran taki sam jak dodawania gatunku do biblioteki, jednak wypełniony danymi. 3. Użytkownik wprowadza zmiany danych. 4. Użytkownik zatwierdza zmiany klikając na odpowiedni przycisk. 5. Ekran edycji zostaje zamknięty.
Przebieg alternatywny	4a. Jeżeli wprowadzone zmiany są nieprawidłowe zostaje wyświetlony stosowny komunikat. Powrót do punktu 3.
Warunki końcowe	<ul style="list-style-type: none"> • Rekord gatunku w bazie danych zostaje zmieniony. • Jeżeli ostatnim widocznym ekranem był ekran danych gatunku, to zostaje on odświeżony i zawiera nowe dane.

Tabela 6.11: PU#011 Usunięcie dodanego przez użytkownika gatunku

PU#011	
Nazwa	Usunięcie dodanego przez użytkownika gatunku
Aktor	Użytkownik
Opis	Użytkownik chce usunąć z biblioteki jeden z dodanych przez siebie gatunków roślin.
Warunki początkowe	<ul style="list-style-type: none"> • Aplikacja jest zainstalowana. • W bazie znajduje się przynajmniej jeden gatunek roślin dodany przez Użytkownika. • Widoczny jest ekran danych gatunku.
Przebieg główny	<ol style="list-style-type: none"> 1. Użytkownik klika na przycisk usunięcia. 2. Wyświetlony zostaje ekran potwierdzenia. 3. Użytkownik kliknięciem w odpowiedni przycisk potwierdza decyzję. 4. Ekran potwierdzenia zostaje zamknięty.
Przebieg alternatywny	3a. W przypadku braku potwierdzenia następuje powrót do punktu 1.
Warunki końcowe	<ul style="list-style-type: none"> • Dane rekordu zostają usunięte z bazy danych łącznie z danymi dotyczącymi powiązanych roślin i przypomnień. • Jeżeli ostatnim widocznym ekranem był ekran danych gatunku wyświetlony zostaje ekran biblioteki, nie zawierający usuniętego rekordu.

Tabela 6.12: Pu#012 Wyświetlenie szczegółów gatunku roślin

PU#012	
Nazwa	Wyświetlenie szczegółów gatunku roślin
Aktor	Użytkownik
Opis	Użytkownik chce wyświetlić szczegółowe dane znajdującego się w bazie gatunku roślin.
Warunki początkowe	<ul style="list-style-type: none"> • Aplikacja jest zainstalowana. • W bazie znajduje się przynajmniej jeden gatunek roślin. • Widoczny jest ekran biblioteki.
Przebieg główny	<ol style="list-style-type: none"> 1. Użytkownik znajduje i klika na wybraną pozycję. 2. Wyświetlony zostaje ekran szczegółów gatunku.
Przebieg alternatywny	
Warunki końcowe	<ul style="list-style-type: none"> • Użytkownik widzi szczegółowe dane wybranego gatunku roślin.

Tabela 6.13: PU#013 Dodanie nowej odmiany do istniejącego gatunku roślin

PU#013	
Nazwa	Dodanie nowej odmiany do istniejącego gatunku roślin
Aktor	Użytkownik
Opis	Użytkownik chce dodać do wybranego gatunku roślin własną odmianę
Warunki początkowe	<ul style="list-style-type: none"> • Aplikacja jest zainstalowana. • Widoczny jest ekran szczegółów gatunku roślin.
Przebieg główny	<ol style="list-style-type: none"> 1. Użytkownik na ekranie szczegółów gatunku roślin znajduje sekcję odmian. 2. Użytkownik znajduje i wybiera przycisk dodawania odmiany. 3. Wyświetlony zostaje ekran dodawania odmiany gatunku. 4. Użytkownik wprowadza nazwę odmiany. 5. Użytkownik wprowadza cechy charakterystyczne odmiany. 6. (Opcjonalne) Użytkownik dodaje zdjęcie odmiany. 7. (Opcjonalne) Użytkownik dodaje notatkę do odmiany. 8. Użytkownik znajduje i wybiera przycisk zatwierdzający wprowadzanie danych. 9. Ekran dodawania odmiany zostaje zamknięty.
Przebieg alternatywny	4a. Jeżeli wybrany gatunek posiada już odmianę o podanej nazwie wyświetlony zostaje stosowny komunikat. Powrót do punktu 4.
Warunki końcowe	<ul style="list-style-type: none"> • Nowy rekord odmiany zostaje dodany do bazy danych. • Jeżeli ostatnim widocznym ekranem był widok szczegółowych danych gatunku zostaje on odświeżony i na liście odmian znajduje się nowy rekord.

Tabela 6.14: PU#014 Edycja dodanej przez użytkownika odmiany roślin

PU#014	
Nazwa	Edycja dodanej przez użytkownika odmiany roślin
Aktor	Użytkownik
Opis	Użytkownik chce wprowadzić zmiany w danych własnoręcznie dodanej odmiany roślin.
Warunki początkowe	<ul style="list-style-type: none"> • Aplikacja jest zainstalowana. • W bazie znajduje się przynajmniej jedna odmiana roślin dodana przez Użytkownika. • Widoczny jest ekran danych odmiany.
Przebieg główny	<ol style="list-style-type: none"> 1. Użytkownik klika na przycisk edycji. 2. Wyświetlony zostaje ekran taki sam jak dodawania odmiany roślin do biblioteki, jednak wypełniony danymi. 3. Użytkownik wprowadza zmiany danych. 4. Użytkownik zatwierdza zmiany klikając na odpowiedni przycisk. 5. Ekran edycji zostaje zamknięty.
Przebieg alternatywny	4a. Jeżeli wprowadzone zmiany są nieprawidłowe zostaje wyświetlony stosowny komunikat. Powrót do punktu 3.
Warunki końcowe	<ul style="list-style-type: none"> • Rekord odmiany w bazie danych zostaje zmieniony. • Jeżeli ostatnim widocznym ekranem był ekran danych odmiany, to zostaje on odświeżony i zawiera nowe dane.

Tabela 6.15: PU#015 Usunięcie dodanej przez użytkownika odmiany roślin

PU#015	
Nazwa	Usunięcie dodanej przez użytkownika odmiany roślin
Aktor	Użytkownik
Opis	Użytkownik chce usunąć z biblioteki jedną z dodanych przez siebie odmian roślin.
Warunki początkowe	<ul style="list-style-type: none"> • Aplikacja jest zainstalowana. • W bazie znajduje się przynajmniej jedna odmiana roślin dodana przez Użytkownika. • Widoczny jest ekran danych odmiany.
Przebieg główny	<ol style="list-style-type: none"> 1. Użytkownik wybiera przycisk usunięcia. 2. Wyświetlony zostaje ekran potwierdzenia. 3. Użytkownik kliknięciem w odpowiedni przycisk potwierdza decyzję. 4. Ekran potwierdzenia zostaje zamknięty.
Przebieg alternatywny	3a. W przypadku braku potwierdzenia następuje powrót do punktu 1.
Warunki końcowe	<ul style="list-style-type: none"> • Dane rekordu zostają usunięte z bazy danych łącznie z danymi dotyczącymi powiązanych roślin i przypomnień. • Jeżeli ostatnim widocznym ekranem był ekran danych wyświetlony zostaje ekran biblioteki, nie zawierający usuniętego rekordu.

Tabela 6.16: PU#016 Wyświetlenie listy posiadanych roślin

PU#016	
Nazwa	Wyświetlenie listy posiadanych roślin
Aktor	Użytkownik
Opis	Użytkownik chce wyświetlić listę wszystkich posiadanych roślin.
Warunki początkowe	<ul style="list-style-type: none"> Aplikacja jest zainstalowana
Przebieg główny	<ol style="list-style-type: none"> Użytkownik klika przycisk listy roślin. Wyświetlony zostaje ekran z listą posiadanych przez niego roślin, posortowana według pokoju oraz alfabetycznie według gatunku.
Przebieg alternatywny	2a. Jeżeli w bazie danych nie ma żadnych posiadanych roślin, to wyświetlony zostaje stosowny komunikat.
Warunki końcowe	<ul style="list-style-type: none"> Użytkownik widzi posortowaną alfabetycznie listę posiadanych przez siebie roślin.

Tabela 6.17: PU#017 Dodanie nowej rośliny do listy posiadanych

PU#017	
Nazwa	Dodanie nowej rośliny do listy posiadanych.
Aktor	Użytkownik
Opis	Użytkownik chce wprowadzić do aplikacji nową roślinę.
Warunki początkowe	<ul style="list-style-type: none"> Aplikacja jest zainstalowana Widoczny jest ekran listy posiadanych roślin
Przebieg główny	<ol style="list-style-type: none"> Użytkownik wybiera przycisk dodania nowej rośliny Wyświetlony zostaje ekran dodawania nowej rośliny Użytkownik dodaje imię rośliny (Opcjonalne) Użytkownika dodaje zdjęcie rośliny Użytkownik wybiera gatunek rośliny z dostępnych opcji Użytkownik wybiera odmianę rośliny z dostępnych opcji Użytkownik wybiera pokój w którym znajduje się roślina. Użytkownik wybiera poziom nasłonecznienia na stanowisku, na którym roślina się znajduje. Użytkownik wybiera czy chce aby został automatycznie utworzony harmonogram przypomnień dla rośliny. Użytkownik znajduje i wybiera przycisk zatwierdzenia. Ekran dodawania rośliny zostaje zamknięty.
Przebieg alternatywny	<ol style="list-style-type: none"> 3a. Jeżeli w bazie znajduje się roślina o tym samym imieniu zostaje wyświetlony stosowny komunikat. Powrót do punktu 3. 7a. Jeżeli użytkownik wybierze opcję „nowy pokój” zostaje uruchomiony PU# Dodawanie nowego pokoju. Następnie powrót do punktu 7. 10a. Jeżeli wprowadzone przez użytkownika dane są nieprawidłowe lub wymagane pola są puste wyświetlony zostaje odpowiedni komunikat. Powrót do punktu 10.
Warunki końcowe	<ul style="list-style-type: none"> Rekord rośliny zostaje dodany do bazy Jeżeli ostatnim widocznym ekranem był ekran listy posiadanych roślin zostaje on odświeżony i widoczny jest nowy rekord.

	<ul style="list-style-type: none"> • Jeżeli Użytkownik wybrał, aby został utworzony harmonogram pielęgnacji dla rośliny dodane do bazy zostaje rekord cyklicznego powiadomienie dotyczącego podlewania.
--	--

Tabela 6.18: PU#018 Edycja istniejącej rośliny

PU#018	
Nazwa	Edycja istniejącej rośliny
Aktor	Użytkownik
Opis	Użytkownik chce wprowadzić zmiany danych istniejącej w bazie danych rośliny
Warunki początkowe	<ul style="list-style-type: none"> • Aplikacja jest zainstalowana • W bazie znajdują się dane przynajmniej jednej rośliny. • Widoczny jest ekran szczegółów wybranej rośliny.
Przebieg główny	<ol style="list-style-type: none"> 1. Użytkownik wybiera przycisk edycji. 2. Wyświetlony zostaje ekran taki sam jak przy dodawaniu nowej rośliny, jednak wypełniony danymi. 3. Użytkownik wprowadza zmiany w danych. 4. Użytkownik zatwierdza zmiany wybierając odpowiedni przycisk. 5. Ekran edycji zostaje zamknięty.
Przebieg alternatywny	4a. Jeżeli zmienione dane są nieprawidłowe wyświetlony zostaje stosowny komunikat. Powrót do punktu 3.
Warunki końcowe	<ul style="list-style-type: none"> • Dane rekordu zostały zaktualizowane w bazie. • Jeżeli ostatnim widocznym ekranem był ekran szczegółów rośliny zostaje on odświeżony i widoczne są nowe dane.

Tabela 6.19: PU#019 Usunięcie istniejącej rośliny

PU#019	
Nazwa	Usunięcie istniejącej rośliny
Aktor	Użytkownik
Opis	Użytkownik chce usunąć z bazy dane wybranej rośliny.
Warunki początkowe	<ul style="list-style-type: none"> • Aplikacja jest zainstalowana • W bazie znajdują się dane przynajmniej jednej rośliny • Widoczny jest ekran szczegółów wybranej rośliny
Przebieg główny	<ol style="list-style-type: none"> 1. Użytkownik wybiera przycisk usunięcia. 2. Wyświetlony zostaje ekran potwierdzenia. 3. Użytkownik wybierając odpowiedni przycisk potwierdza decyzję. 4. Ekran potwierdzenia zostaje zamknięty.
Przebieg alternatywny	3a. W przypadku braku potwierdzenia następuje powrót do punktu 1.
Warunki końcowe	<ul style="list-style-type: none"> • Dane rekordu zostają usunięte z bazy danych łącznie z danymi dotyczącymi powiązanych przypomnień. • Jeżeli ostatnim widocznym ekranem był ekran danych rośliny wyświetlony zostaje odświeżony ekran listy roślin, nie zawierający usuniętego rekordu.

Tabela 6.20: PU#020 Wyświetlenie danych rośliny

PU#020	
Nazwa	Wyświetlenie danych rośliny
Aktor	Użytkownik
Opis	Użytkownik chce wyświetlić szczegółowe dane rośliny znajdującej się w bazie.
Warunki początkowe	<ul style="list-style-type: none"> • Aplikacja jest zainstalowana • Widoczny jest ekran listy posiadanych roślin • W bazie danych znajduje się przynajmniej jeden rekord rośliny
Przebieg główny	<ol style="list-style-type: none"> 1. Użytkownik znajduje szukaną pozycję i klika na nią. 2. Wyświetlony zostaje ekran danych rośliny.
Przebieg alternatywny	
Warunki końcowe	<ul style="list-style-type: none"> • Użytkownik widzi dane wybranej rośliny.

Tabela 6.21: PU#021 Wyświetlenie ustawień aplikacji

PU#021	
Nazwa	Wyświetlenie ustawień aplikacji
Aktor	Użytkownik
Opis	Użytkownik chce zobaczyć ustawienia aplikacji.
Warunki początkowe	<ul style="list-style-type: none"> • Aplikacja jest zainstalowana • Widoczny ekran listy aktywnych przypomnień
Przebieg główny	<ol style="list-style-type: none"> 1. Użytkownik znajduje i wybiera przycisk z ikoną ustawień 2. Wyświetlony zostaje ekran ustawień aplikacji
Przebieg alternatywny	
Warunki końcowe	<ul style="list-style-type: none"> • Użytkownik widzi ekran ustawień aplikacji

Tabela 6.22: Pu#022 Założenie konta użytkownika

PU#022	
Nazwa	Założenie konta użytkownika
Aktor	Użytkownik
Opis	Użytkownik chce założyć konto.
Warunki początkowe	<ul style="list-style-type: none"> • Aplikacja jest zainstalowana • Widoczny ekran ustawień aplikacji
Przebieg główny	<ol style="list-style-type: none"> 1. Użytkownik znajduje i wybiera przycisk logowania. 2. Wyświetlony zostaje ekran logowania. 3. Użytkownik znajduje i wybiera przycisk zakładania nowego konta. 4. Wyświetlony zostaje ekran zakładania nowego konta. 5. Użytkownik wprowadza swoje imię. 6. Użytkownik wprowadza adres email. 7. Użytkownik wprowadza hasło.

	8. Użytkownik znajduje i wybiera przycisk potwierdzający chęć założenia konta. 9. Wyświetlony zostaje komunikat o powodzeniu założenia konta. 10. Ekran zakładania konta zostaje zamknięty.
Przebieg alternatywny	6a. Jeżeli wprowadzony adres email jest nieprawidłowy lub na podany adres zostało już założone konto wyświetlany jest odpowiedni komunikat. Powrót do punktu 5. 7a. Jeżeli wprowadzone hasło jest nieprawidłowe zostaje wyświetlony odpowiedni komunikat. Powrót do punktu 6. 9a. W przypadku niepowodzenia zostaje wyświetlony odpowiedni komunikat o błędzie.
Warunki końcowe	<ul style="list-style-type: none"> • Rekord konta użytkownika zostaje dodany do bazy danych.

Tabela 6.23: PU#023 Zalogowanie

PU#023	
Nazwa	Zalogowanie
Aktor	Użytkownik
Opis	Użytkownik posiadający konto chce się zalogować w aplikacji
Warunki początkowe	<ul style="list-style-type: none"> • Aplikacja jest zainstalowana • Użytkownik posiada konto w aplikacji • Widoczny ekran ustawień aplikacji
Przebieg główny	1. Użytkownik znajduje i wybiera przycisk logowania. 2. Użytkownik wprowadza adres email. 3. Użytkownik wprowadza hasło. 4. Użytkownik znajduje i wybiera przycisk potwierdzający logowanie w aplikacji.
Przebieg alternatywny	2a. Jeżeli w bazie danych aplikacji nie istnieje konto utworzone z pomocą podanego adresu email wyświetlony zostaje stosowny komunikat. Powrót do punktu 2. 3a. Jeżeli podane hasło jest nieprawidłowe wyświetlony zostaje stosowny komunikat. Powrót do punktu 3. 3b. Jeżeli Użytkownik nie pamięta hasła, to znajduje i wybiera odpowiedni przycisk i na jego adres mailowy zostaje automatycznie wysłana wiadomość z obecnie obowiązującym hasłem. Powrót do punktu 3.
Warunki końcowe	<ul style="list-style-type: none"> • Użytkownik jest teraz Użytkownikiem Zalogowanym • Po powrocie na stronę główną na górze ekranu będzie wyświetlane jego imię.

Tabela 6.24: PU#024 Zapis kopii danych na serwerze

PU#024	
Nazwa	Zapis kopii danych na serwerze
Aktor	Użytkownik Zalogowany (UZ)
Opis	Użytkownik Zalogowany chce zapisać kopię zapasową danych na serwerze.

Warunki początkowe	<ul style="list-style-type: none"> • Aplikacja jest zainstalowana. • Widoczny jest ekran ustawień aplikacji.
Przebieg główny	<ol style="list-style-type: none"> 1. UZ znajduje i wybiera przycisk wykonywania kopii danych na serwerze. 2. Wyświetlony zostaje komunikat o pomyślnym wykonaniu operacji.
Przebieg alternatywny	<ol style="list-style-type: none"> 2a. W przypadku niepowodzenia wykonania operacji zostaje wyświetlony odpowiedni komunikat.
Warunki końcowe	<ul style="list-style-type: none"> • W zewnętrznej bazie danych rekord Konta Użytkownika zostaje zaktualizowany o kopię danych znajdujących się lokalnie w aplikacji. • Ekran ustawień aplikacji zostaje odświeżony i zawiera przycisk przywrócenia danych z kopii.

Tabela 6.25: PU#025 Przywracanie danych z kopii

PU#025	
Nazwa	Przywracanie danych z kopii
Aktor	Użytkownik Zalogowany (UZ)
Opis	Użytkownik Zalogowany chce przywrócić dane z kopii na serwerze do lokalnej instancji aplikacji.
Warunki początkowe	<ul style="list-style-type: none"> • Aplikacja jest zainstalowana • Widoczny jest ekran ustawień aplikacji • Rekord Konta Użytkownika w bazie danych zawiera uprzednio wykonaną kopię danych.
Przebieg główny	<ol style="list-style-type: none"> 1. UZ znajduje i wybiera przycisk przywracania danych z kopii z serwera. 2. Wyświetlony zostaje ekran zapytania o potwierdzenie wykonania operacji. 3. UZ potwierdza operację. 4. Wyświetlony zostaje komunikat o pomyślnym wykonaniu operacji.
Przebieg alternatywny	<ol style="list-style-type: none"> 3a. W przypadku braku potwierdzenia przez UZ następuje powrót do punktu 1. 4a. W przypadku niepowodzenia zostaje wyświetlony stosowny komunikat.
Warunki końcowe	<ul style="list-style-type: none"> • Dane w lokalnej bazie danych aplikacji zostają zastąpione danymi znajdującymi się na serwerze. • Aplikacja zostaje uruchomiona ponownie. UZ widzi dane pobrane z serwera.

Tabela 6.26: PU#026 Wylogowanie

PU#026	
Nazwa	Wylogowanie
Aktor	Użytkownik Zalogowany (UZ)
Opis	Użytkownik Zalogowany chce się wylogować.
Warunki początkowe	<ul style="list-style-type: none"> • Aplikacja jest zainstalowana • Widoczny jest ekran ustawień aplikacji

Przebieg główny	<ol style="list-style-type: none"> 1. UZ znajduje i wybiera przycisk wylogowania. 2. Zostaje wyświetlony ekran z komunikatem o potwierdzeniu wylogowania.
Przebieg alternatywny	2a. W przypadku niepowodzenia wyświetlony zostaje stosowny komunikat. Powrót do punktu 1.
Warunki końcowe	<ul style="list-style-type: none"> • Użytkownik Zalogowany zostaje wylogowany. • Ekran z komunikatem zostaje zamknięty. • Ekran ustawień aplikacji zostaje odświeżony. Nie zawiera danych Użytkownika, za to zawiera przycisk logowania.

Tabela 6.27: PU#027 Wyświetlenie listy życzeń

PU#027	
Nazwa	Wyświetlenie listy życzeń.
Aktor	Użytkownik
Opis	Użytkownik chce wyświetlić swoją listę życzeń.
Warunki początkowe	<ul style="list-style-type: none"> • Aplikacja jest zainstalowana. • Widoczny jest ekran główny aplikacji.
Przebieg główny	<ol style="list-style-type: none"> 1. Użytkownik znajduje i wybiera przycisk listy życzeń. 2. Wyświetlony zostaje ekran listy życzeń.
Przebieg alternatywny	
Warunki końcowe	<ul style="list-style-type: none"> • Jeżeli lista życzeń jest pusta, na ekranie widoczny jest odpowiedni komunikat. • Użytkownik widzi listę życzeń.

Tabela 6.28: PU#028 Dodanie odmiany roślin do listy życzeń

PU#028	
Nazwa	Dodanie odmiany roślin do listy życzeń
Aktor	Użytkownik
Opis	Użytkownik chce dodać odmianę roślin do listy życzeń.
Warunki początkowe	<ul style="list-style-type: none"> • Aplikacja jest zainstalowana. • Widoczny jest ekran szczegółów gatunku w bibliotece gatunków.
Przebieg główny	<ol style="list-style-type: none"> 1. Użytkownik znajduje interesującą go odmianę i wybiera ikonę serca znajdującą się po prawej stronie nazwy odmiany. 2. Wybrana ikona zmienia się na wypełnioną oznaczając w ten sposób odmianę znajdującą się już na liście życzeń.
Przebieg alternatywny	2a. W przypadku niepowodzenia dodawania odmiany do listy życzeń wyświetlony zostaje stosowny komunikat.
Warunki końcowe	<ul style="list-style-type: none"> • Odmiana roślin zostaje dodana do listy życzeń Użytkownika.

Tabela 6.29: PU#029 Usunięcie odmiany roślin z listy życzeń

PU#029	
Nazwa	Usunięcie odmiany roślin z listy życzeń
Aktor	Użytkownik
Opis	Użytkownik chce usunąć odmianę roślin z listy życzeń.
Warunki początkowe	<ul style="list-style-type: none"> • Aplikacja jest zainstalowana • Na liście życzeń Użytkownik znajduje się przynajmniej jedna odmiana roślin. • Widoczny jest ekran szczegółów gatunku w bibliotece albo ekran listy życzeń Użytkownik.
Przebieg główny	<ol style="list-style-type: none"> 1. Użytkownik znajduje interesującą go odmianę i wybiera wypełnioną ikonę serca znajdującą się po prawej stronie od nazwy odmiany. 2. Wyświetlony zostaje komunikat o pomyślnym wykonaniu operacji.
Przebieg alternatywny	2a. W przypadku niepowodzenia wykonania operacji zostaje wyświetlony odpowiedni komunikat.
Warunki końcowe	<ul style="list-style-type: none"> • Wybrana odmiana roślin zostaje usunięta z listy życzeń Użytkownika. • Jeżeli ostatnim widocznym ekranem był ekran listy życzeń zostaje on odświeżony i nie zawiera usuniętej odmiany roślin. • Jeżeli ostatnim widocznym ekranem był ekran szczegółów gatunku to zostaje on odświeżony i ikona serca znajdującą się po prawej stronie szczegółów gatunku jest teraz pusta.

6.3. Baza danych

W niniejszym podrozdziale zostanie opisany zarys projektu lokalnej bazy danych aplikacji Planter.

6.3.1. Room

Room jest częścią Android Jetpack, zbioru bibliotek mającego na celu ułatwienie programistom wdrażania właściwych praktyk i pomagająca w utrzymaniu standardów kodowania na przestrzeni różnych wersji oprogramowania i sprzętu. Ta konkretna biblioteka stanowi warstwę abstrakcji nad SQLite zapewniając łatwiejszy dostęp do danych m.in. poprzez automatyczną konwersję między postacią tabel SQLite i obiektami Java, czy też oferując różne rozwiązania w zakresie przetwarzania danych.

Na ten moment (styczeń 2022) jest to rekomendowany przez Google sposób obsługi i implementacji baz danych w systemie Android, zamiast korzystania z czystego SQLite.

6.3.2. Relacje

Dzięki zastosowaniu wyżej wspomnianej biblioteki Room nie ma konieczności tworzenia klasycznych relacji na zasadzie kluczy obcych. Jest to możliwe, ponieważ Room zapewnia sposób na określenie, które pola odnoszą się do innego obiektu (np. za pomocą znacznika @Embedded). Poniżej, takie odniesienia zaznaczone są *kursywą*, a w nawiasie znajduje się informacja skąd pochodzi dane pole.

1. **Plant**(PLANT_ID, NAME, DESCRIPTION, LAST_WATER, IN_GARDEN_START, *ROOM_ID*(Room), *VARIETY_ID*(Variety))
2. **Species**(SPECIES_ID, NAME, SCIENTIFIC_NAME, DETAILS, SUN_LEVEL, HUMIDITY, DIFFICULTY, USER_ADDED)
3. **Variety**(VARIETY_ID, NAME, USER_ADDED, IN_FAVORITES, *SPECIES_ID*(Species))
4. **Room**(ROOM_ID, NAME)
5. **Notification**(NOTIFICATION_ID, NAME, TYPE, STARTING_DATE, CLOSING_DATE, IS_REPEATABLE, PERIOD, *PLANT_ID*(Plant))

6.3.3. Schemat bazy danych

Przedstawiony na rysunku 6.2 schemat bazy danych ma zadanie obrazować rodzaje i liczbę powiązań i relacji między poszczególnymi encjami w projektowanej bazie danych.



Rysunek 6.2: Schemat bazy danych aplikacji Planter. Źródło: opracowanie własne.

6.4. Diagram Klas

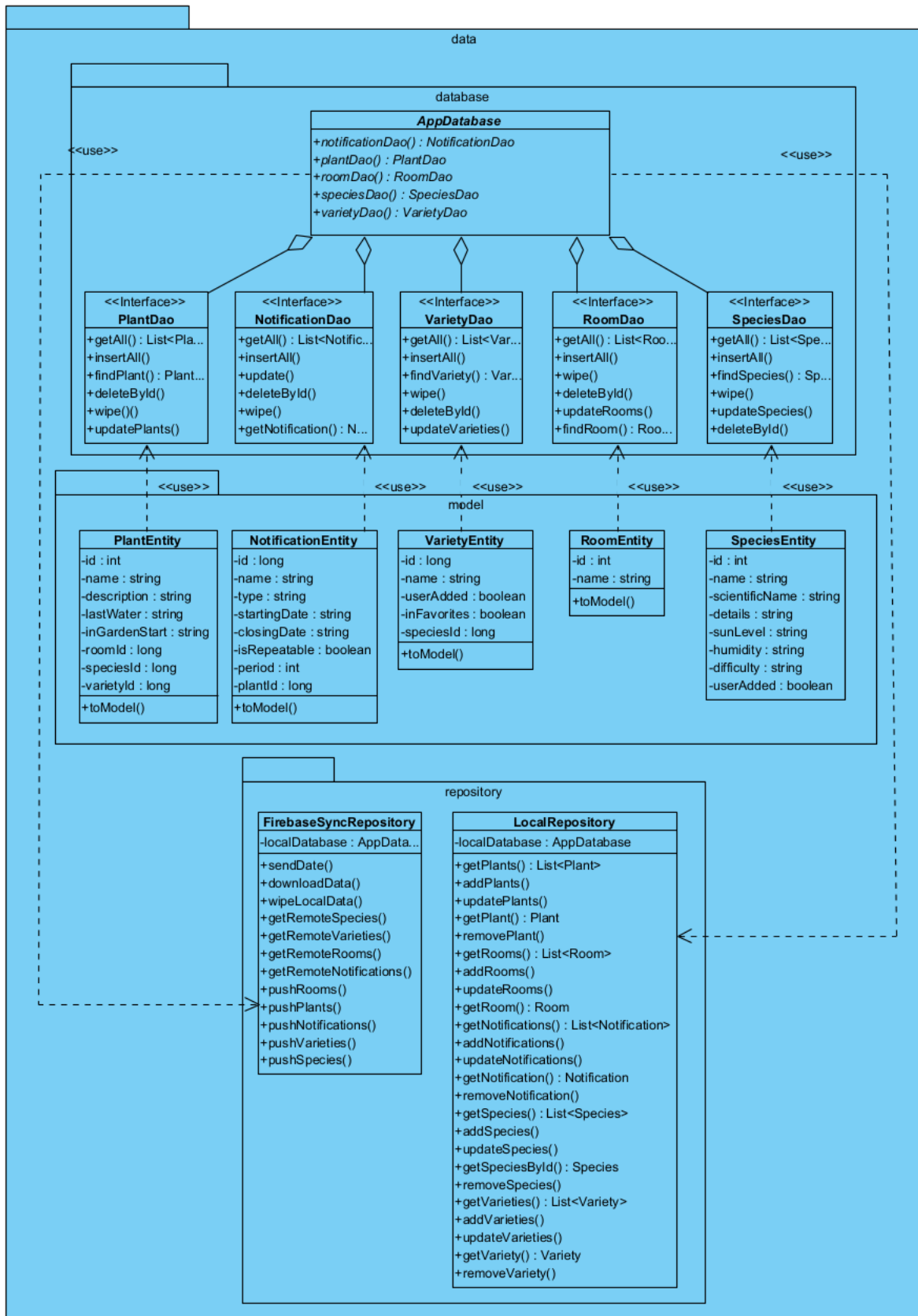
W odniesieniu do przedstawionego powyżej schematu bazy danych, diagram klas obrazuje strukturę pakietów i klas całego projektu. W celu zwiększenia czytelności diagramu podzieliłam go na trzy pakiety – pakiet data (rysunek 6.3), pakiet domain (rysunek 6.4) oraz pakiet presentation (rysunek 6.5). Analogicznie, w ramach aplikacji Planter zastosowana zostanie architektura trójwarstwowa.

Klasy pakietu data odpowiedzialne są za bezpośrednie interakcje aplikacji z bazą danych. Zarówno lokalną, jak i zewnętrzną przechowującą dane użytkowników. W ramach tego pakietu dane są bezpośrednią reprezentacją encji bazy danych.

Klasy pakietu domain mają za zadanie przekształcanie danych z postaci reprezentacji encji (data class) do postaci zwykłych klas języka Kotlin i na odwrót. W tej „warstwie” określone są również wszystkie typy wyliczeniowe pomagające ograniczać wartości mogące znaleźć się w bazie, oraz obsługiwane są powiadomienia i autoryzacja użytkownika.

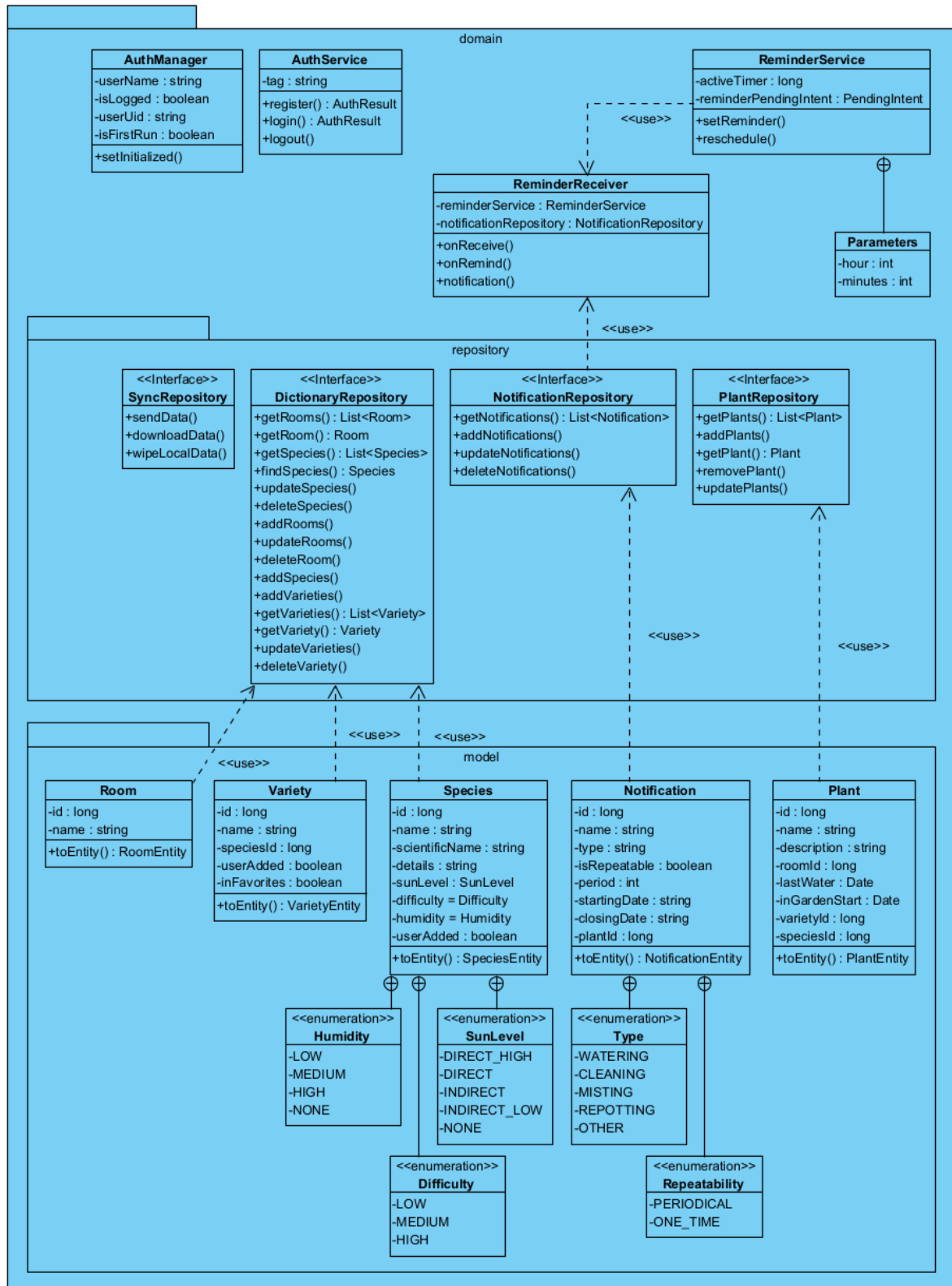
Ostatecznie klasy pakietu presentation odpowiadają za komunikację aplikacji z interfejsem użytkownika, np. wiążąc jego poszczególne elementy z wywołaniami konkretnych metod klas. Wewnątrz tego pakietu znajdują się również reprezentacje danych spreparowane pod kątem konkretnych widoków aplikacji.

6.4.1. Diagram klas pakietu data



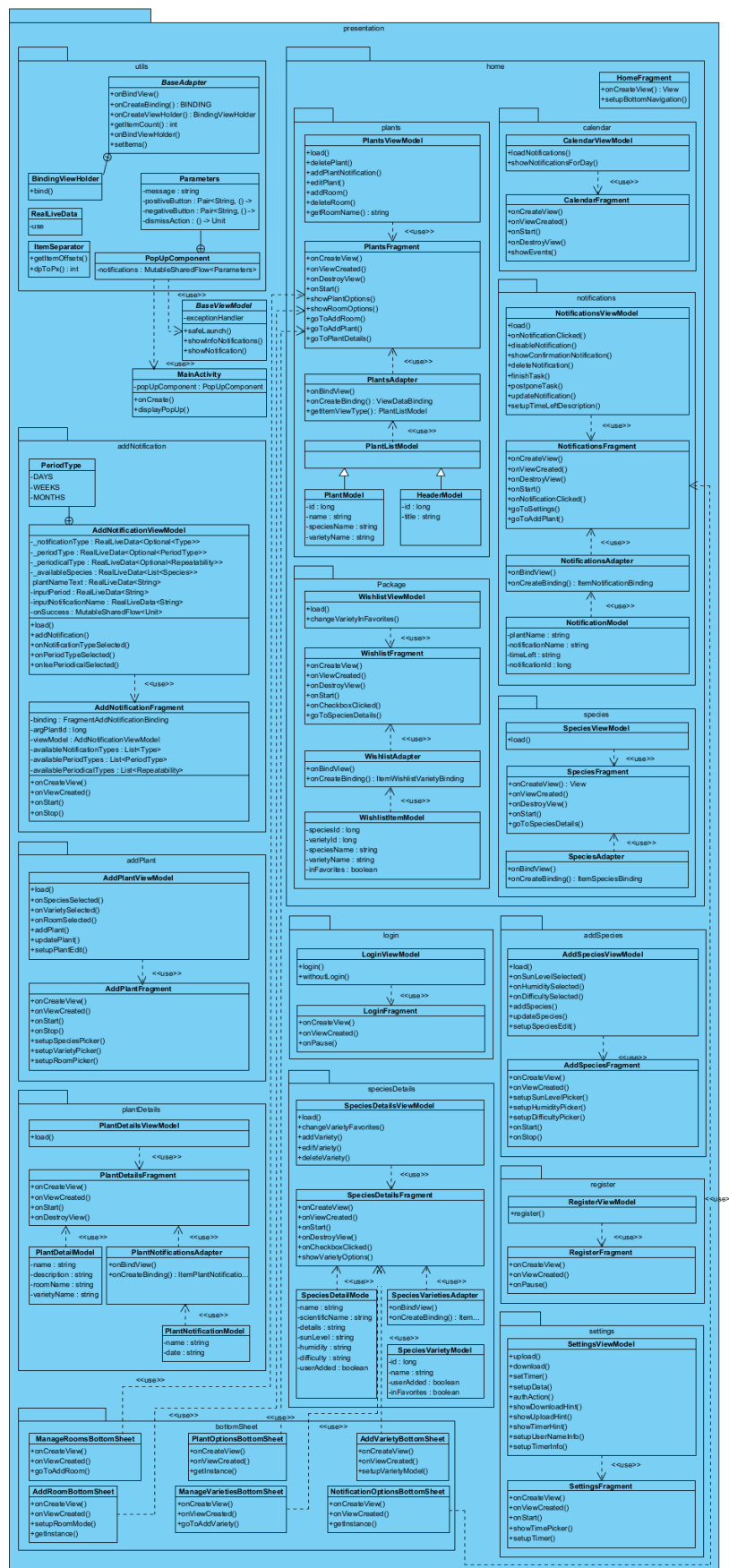
Rysunek 6.3: Diagram klas pakietu data. Źródło: opracowanie własne

6.4.2. Diagram klas pakietu domain



Rysunek 6.4: Diagram klas pakietu domain. Źródło: opracowanie własne

6.4.3. Diagram klas pakietu presentation



Rysunek 6.5: Diagram klas pakietu presentation. Źródło: opracowanie własne

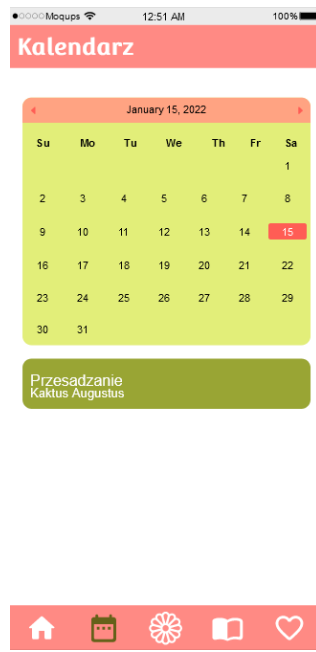
6.5. Projekt interfejsu

Wykonanie projektu interfejsu użytkownika pozwala określić jakie funkcjonalności powinna zapewniać aplikacja, oraz ich rozmieszczenie na poszczególnych ekranach. Projekt interfejsu aplikacji Planter obejmuje pięć podstawowych ekranów dostępnych z poziomu dolnego paska:

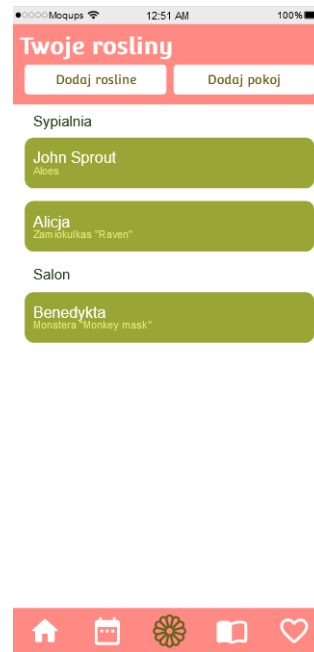
1. Ekran aktywnych powiadomień (Rysunek 6.6),
2. Ekran widoku kalendarza (Rysunek 6.7),
3. Ekran posiadanych roślin (Rysunek 6.8),
4. Ekran biblioteki gatunków (Rysunek 6.9),
5. Ekran listy życzeń (Rysunek 6.10).



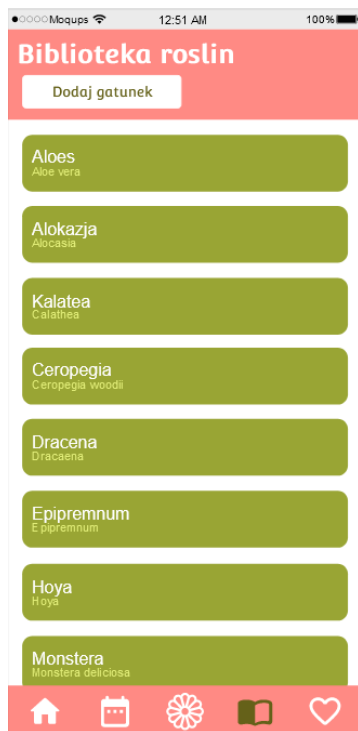
Rysunek 6.6: Projekt ekranu listy aktywnych powiadomień.
Źródło: opracowanie własne



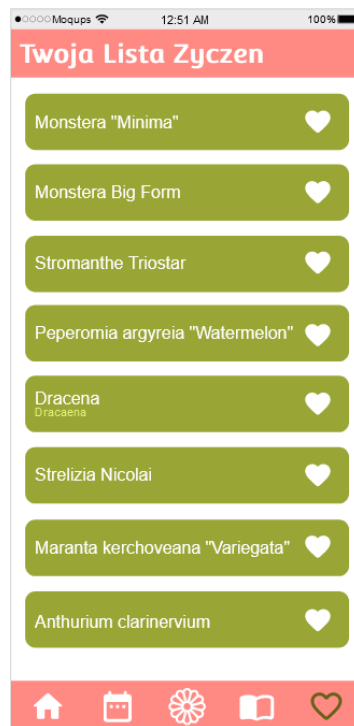
Rysunek 6.7: Projekt ekranu widoku kalendarza. Źródło: opracowanie własne



Rysunek 6.8: Projekt ekranu widoku posiadanych roślin.
Źródło: opracowanie własne



Rysunek 6.9: Projekt ekranu widoku biblioteki gatunków roślin. Źródło: opracowanie własne



Rysunek 6.10: Projekt ekranu widoku listy życzeń. Źródło: opracowanie własne

Poza przedstawionymi ekranami zaimplementowane zostaną również:

6. Ekran logowania/rejestracji,
7. Ekran ustawień,
8. Ekran dodawania/edycji gatunku,
9. Ekran dodawania/edycji rośliny.

7. Implementacja

7.1. Wstęp

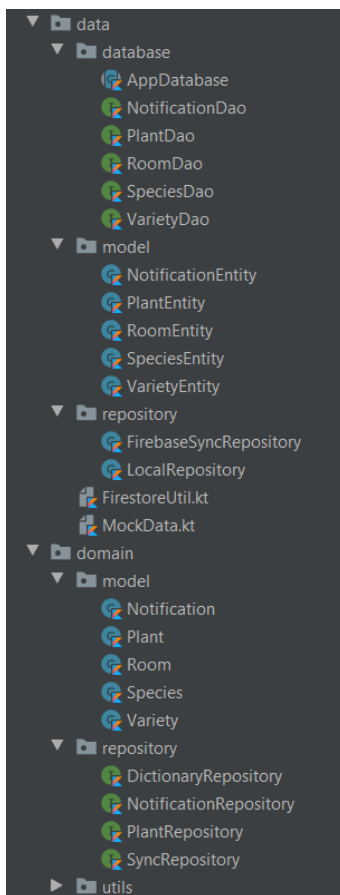
W poniższym rozdziale wykonany zostanie opis implementacji aplikacji Planter. Uwaga w tym zakresie zostanie poświęcona przede wszystkim strukturze projektu aplikacji, wykorzystanym wzorcom projektowym, problemom napotkanym na drodze implementacji, przeprowadzonym testom oraz ekranom gotowej aplikacji. Krótko opisane zostanie również środowisko, w którym powstawało rozwiązanie.

7.2. Środowisko

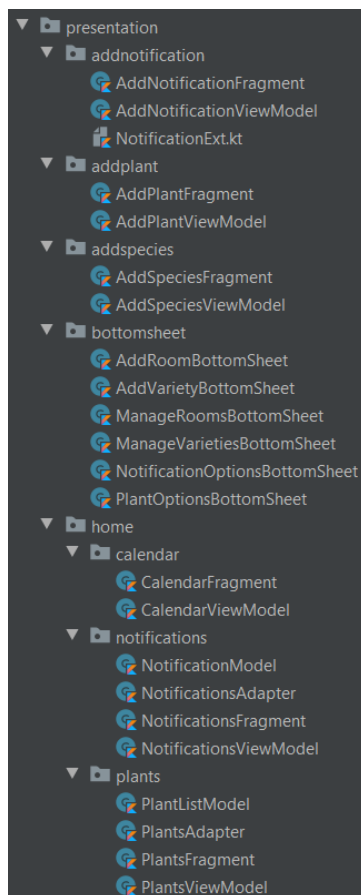
Aplikacja Planter została zaimplementowana w środowisku Android Studio (wersja 4.1.2), na komputerze przenośnym marki Lenovo z systemem operacyjnym Windows 10.

7.3. Elementy projektu

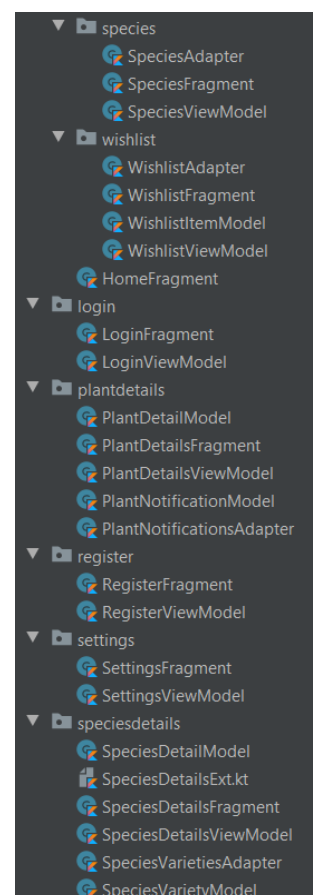
Dwoma bardzo istotnymi dla zrozumienia struktury projektu elementami są diagramy pakietów i zasobów. Pierwszy z nich obrazuje strukturę pakietów i klas w projekcie, drugi zaś dostarcza informacji na temat interfejsów użytkownika i ich pomniejszych elementów. Na rysunkach 7.1-7.3 przedstawiony jest diagram pakietów projektu aplikacji Planter. Razem, oba wspomniane wyżej elementy stanowią podstawę struktury projektu, z której narzędzie Gradle tworzy plik pozwalający zainstalować aplikację na urządzeniu przenośnym lub emulatorze.



Rysunek 7.1: Diagram pakietów, część pierwsza. Źródło: opracowanie własne



Rysunek 7.2: Diagram pakietów, część druga. Źródło: opracowanie własne



Rysunek 7.3: Diagram pakietów, część trzecia. Źródło: opracowanie własne

Oprócz diagramów pakietów i zasobów projekt obejmuje również pliki konfiguracyjne oraz pakiety testów jednostkowych aplikacji. Poszczególne pakiety zostały krótko opisane w Tabeli 7.1.

Tabela 7.1: Pakiety aplikacji Planter

Lp.	Nazwa pakietu	Opis
1	data	Pakiet zawiera definicję struktury bazy danych, wliczając reprezentację encji, obiekty dostępu do danych, implementację metod repozytoriów zdefiniowanych w pakiecie domain, oraz implementację komunikacji z lokalną i zewnętrzną bazą danych.
2	data.database	Pakiet zawiera definicje obiektów dostępu do danych.
3	data.model	Pakiet zawiera definicję encji bazy danych.
4	data.repository	Pakiet zawiera definicję metod używanych przez wyższe warstwy oprogramowania, służących do komunikacji z zewnętrzną i lokalną bazą danych.
5	domain	Pakiet odpowiada za umożliwianie i ułatwianie wykonywania operacji na danych z poziomu warstwy prezentacji oraz za obsługę powiadomień.
6	domain.model	Pakiet zawiera definicje obiektów znajdujących się w bazie danych w postaci zwykłych klas.
7	domain.repository	Pakiet zawiera interfejsy umożliwiające wykonywanie operacji na danych z poziomu warstwy prezentacji.
8	domain.utils	Pakiet zawiera metody i struktury pomocnicze dla pozostałych pakietów i klas pakietu domain.
9	presentation	Pakiet odpowiada za obsługę wyświetlania poszczególnych ekranów aplikacji.
10	presentation.addnotification	Pakiet odpowiada za wyświetlanie i obsługę informacji i poleceń otrzymywanych za pomocą ekranu dodawania powiadomień.
11	presentation.addplant	Pakiet odpowiada za wyświetlanie i obsługę informacji i poleceń otrzymywanych za pomocą ekranu dodawania roślin.
12	presentation.addspecies	Pakiet odpowiada za wyświetlanie i obsługę informacji i poleceń otrzymywanych za pomocą ekranu dodawania gatunków.
13	presentation.bottomsheet	Pakiet odpowiada za wyświetlanie i obsługę informacji i poleceń otrzymywanych za pomocą wszystkich mniejszych dolnych fragmentów aplikacji (dotyczących np. wyboru opcji edycji/usunięcia obiektu, lub dodawania prostych obiektów np. pokoju).
14	presentation.home	Pakiet odpowiada za wyświetlanie i obsługę poleceń otrzymywanych za pomocą wszystkich pięciu fragmentów składających się na ekran domowy (powiadomienia, kalendarz, rośliny, biblioteka, lista życzeń).
15	presentation.login	Pakiet odpowiada za wyświetlanie i obsługę informacji i poleceń otrzymywanych za pomocą ekranu logowania.

16	presentation.plantdetails	Pakiet odpowiada za wyświetlanie i obsługę poleceń otrzymywanych za pomocą ekranu szczegółów rośliny.
17	presentation.register	Pakiet odpowiada za wyświetlanie i obsługę informacji i poleceń otrzymywanych za pomocą ekranu rejestracji użytkownika.
18	presentation.settings	Pakiet odpowiada za wyświetlanie i obsługę poleceń otrzymywanych za pomocą ekranu ustawień.
19	presentation.speciesDetails	Pakiet odpowiada za wyświetlanie i obsługę poleceń otrzymywanych za pomocą ekranu szczegółów gatunku roślin.
20	presentation.utils	Pakiet zawiera metody i struktury pomocnicze dla pozostałych pakietów.

7.4. Wzorce projektowe

Wzorce projektowe to ogólnie przyjęte typowe rozwiązania problemów, które często pojawiają się w trakcie projektowania oprogramowania oraz implementacji. W przeciwieństwie do algorytmów nie zapewniają one jednak konkretnych wytycznych, np. w postaci następujących po sobie kroków, jedynie uogólnione koncepcje rozwiązań.

Wyróżnia się trzy podstawowe kategorie wzorców projektowych. Kreacyjne odnoszą się do sposobów i okoliczności tworzenia instancji obiektów, strukturalne dotyczą metod organizacji elementów projektu, zaś behawioralne opisują to jak poszczególne komponenty oddziałują i komunikują się między sobą [26].

7.4.1. Adapter

Adapter należy do kategorii strukturalnych wzorców projektowych i pozwala on na współdziałanie ze sobą obiektów o niekompatybilnych interfejsach poprzez dodanie elementu pośredniczącego [27]. W ramach implementacji aplikacji Planter zostało zastosowanych wiele adapterów. Zadaniem większości z nich jest umożliwienie wyświetlania danych z bazy w postaci listy, a to znaczące podobieństwo w sposobie ich działania umożliwiło wydzielenie ich wspólnej części do abstrakcyjnej klasy BaseAdapter, znajdującej się w pakiecie presentation.utils.

7.4.2. Mediator

Mediator zalicza się do behawioralnych wzorców projektowych. Jego założeniem jest zredukowanie bezpośredniej komunikacji między tymi komponentami projektu, które mają pozostać niezależne oraz uproszczenie tej komunikacji [28]. W aplikacji Planter wzorzec ten został zastosowany w pakiecie domain, za pomocą interfejsów pośredniczących w komunikacji pomiędzy bazą danych a warstwą obsługującą interfejs użytkownika.

7.4.3. Wstrzykiwanie zależności

Wstrzykiwanie zależności wymiga się klasyfikacjom według wymienionych wyżej kategorii. Wzorzec ten polega na odwróceniu zależności, czyli na zdjęciu z obiektu odpowiedzialności za tworzenie wewnętrznego obiektu, na rzecz otrzymania gotowej instancji na przykład poprzez konstruktor [29]. Na przestrzeni implementacji aplikacji Planter jest on bardzo szeroko stosowany, przede wszystkim w warstwie prezentacji. Poza wstrzykiwaniem przez konstruktor został również wykorzystany framework Koin. Znalazł on zastosowanie przy obsłudze powiadomień w klasach ReminderReceiver oraz MainActivity.

7.5. Implementacja bazy danych

Implementacja bazy danych składa się z trzech podstawowych elementów. Listing 7.1 przedstawia fragment kodu służący zbudowaniu bazy danych i wypełnieniu jej wstępnymi danymi.

```
val databaseModule = module { this: Module
    single { this: Scope
        Room.databaseBuilder(
            androidContext(),
            AppDatabase::class.java,
            name: "local-database"
        )
        .createFromAsset( databaseFilePath: "database/default_data.db")
        .build()
    }
}
```

Listing 7.1: Kod w języku Kotlin inicjujący bazę danych

Korzystanie z biblioteki Room do implementacji bazy danych wymaga zdefiniowania encji, przykładowy kod encji gatunku roślin przedstawiony jest na Listingu 7.2.

```
@Entity(tableName = AppDatabase.DB_SPECIES)
data class SpeciesEntity(
    @PrimaryKey(autoGenerate = true) val id: Long? = null,
    val name: String? = null,
    val scientificName: String? = null,
    val details: String? = null,
    val sunLevel: String? = null,
    val humidity: String? = null,
    val difficulty: String? = null,
    val userAdded: Boolean? = false,
) {
    companion object {
        fun SpeciesEntity.toModel(): Species =
            Species(
                id = id ?: error("Species with empty ID!"),
                name = name ?: error("Species with empty NAME!"),
                scientificName = scientificName,
                details = details ?: error("Species with empty DETAILS!"),
                sunLevel = sunLevel?.let(Species.SunLevel::valueOf) ?: error("Species with empty SUN LEVEL!"),
                humidity = humidity?.let(Species.Humidity::valueOf) ?: error("Species with empty HUMIDITY!"),
                difficulty = difficulty?.let(Species.Difficulty::valueOf) ?: error("Species with empty DIFFICULTY!"),
                userAdded = userAdded ?: error("Species with empty USER_ADDED!"),
            )
    }
}
```

Listing 7.2: Kod w języku Kotlin definiujący encję bazy danych (przykład)

Aby móc korzystać z danych reprezentowanych przez uprzednio zdefiniowane encje biblioteki Room należy zaimplementować obiekty dostępu do danych (przykład na Listingu 7.3).

```
@Dao
interface RoomDao {

    @Query( value: "SELECT * FROM ${AppDatabase.DB_ROOMS}")
    suspend fun getAll(): List<RoomEntity>

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insertAll(vararg rooms: RoomEntity)

    @Query( value: "DELETE FROM ${AppDatabase.DB_ROOMS}")
    suspend fun wipe()

    @Query( value: "DELETE FROM ${AppDatabase.DB_ROOMS} WHERE id = :id")
    fun deleteById(id: Long)

    @Update
    fun updateRooms(vararg entities: RoomEntity)

    @Query( value: "SELECT * FROM ${AppDatabase.DB_ROOMS} WHERE id = :roomId")
    fun findRoom(roomId: Long): RoomEntity
}
```

Listing 7.3 Kod w języku Kotlin definiujący metody obiektu dostępu do danych (przykład)

Komunikacja z zewnętrzną bazą danych w serwisie Firebase została zaimplementowana w pliku `FirebaseSyncRepository.kt` w pakiecie `data.repository`.

7.6. Problemy implementacyjne

Realizacja jakiegokolwiek projektu wiąże się z napotkaniem po drodze różnego rodzaju problemów, których przewidzenie na etapie projektowania jest niemożliwe. Problemy te mogą być natury technicznej, lub wynikać z błędów w projekcie. Również podczas implementacji niniejszej aplikacji nie udało się tego ominąć. Poniżej w tabeli 7.2 opisany został najbardziej znaczący problemy, który został napotkany podczas realizacji projektu.

Tabela 7.2: PI#001 Wyświetlenie przypomnień ustawionych przez użytkownika w formie kalendarza

PI#001	
Nazwa	Wyświetlanie przypomnień ustawionych przez użytkownika w formie kalendarza.
Opis	W związku z chęcią wdrożenia w aplikacji Planter założeń Material Design, założyłam skorzystanie z implementacji fragmentu kalendarza oferowanej przez Material Design Components. Niestety okazało się, że oferowany widok kalendarza nie spełniał potrzeb projektu ze względu na to, że zapewnienie zaprojektowanych funkcjonalności wymagałoby połączenie <code>CalendarPicker</code> służącego do dokonywania przez użytkownika wyboru daty lub przedziału dat z klasycznym widokiem kalendarza aby móc umieścić ikony pod dniami, na które zaplanowane jest powiadomienie.
Rozwiązanie	W celu usprawnienia pracy nad implementacją aplikacji skorzystałam z gotowego rozwiązania udostępnianego na platformie GitHub pod

	licencją open-source w postaci gotowego widgeta łączącego funkcjonalności obu widoków oferowanych przez Material Design Components. Rozwiązanie to nosi nazwę MaterialCalendarView i jest oferowane przez firmę applandeo [źródło]. Zgodnie z nazwą, rozwiązanie to opiera się również na założeniach Material Design zapewniając ciągłość wizualną interfejsu.
--	---

7.7. Testy

W podstawowym ujęciu testy oprogramowania dzielą się na dwa rodzaje: testy lokalne i testy instrumentacyjne. Testy lokalne dotyczą tych części kodu, które do swojego działania nie wymagają korzystania z docelowego sprzętu ani API. Z reguły dotyczą one pewnych wewnętrznych obliczeń lub zadań logiki biznesowej. Testy instrumentacyjne z kolei, mogą już wymagać korzystania ze sprzętu, emulatora lub API. W niniejszym podrozdziale zostaną opisane testy, które zostały przeprowadzone w drodze implementacji aplikacji Planter.

7.7.1. Testy jednostkowe

Testy jednostkowe należą do testów lokalnych. Polegają one na testowaniu jak najmniejszej możliwej do logicznego wydzielenia części kodu, co w praktyce oznacza zazwyczaj pewną funkcję lub metodę. Może ona dotyczyć zarówno pewnych obliczeń wykonywanych przez aplikację, jak również na przykład operacji wykonywanych na lokalnej bazie danych. W implementowanej aplikacji dobrym miejscem do przeprowadzenia testów lokalnych jest pakiet domain.utils i zawarte w nim funkcje pomocnicze dotyczące przekształcania postaci dat oraz obliczania rozpiętości przedziału dat. Listing 7.4 przedstawia jeden z testów jednostkowych zaimplementowanych przy użyciu biblioteki JUnit.

```
@RunWith(value = Parameterized::class)
class DateTimeUtilsDaysBetweenTest(
    private val dateFromString: String,
    private val dateToString: String,
    private val expected: Int
) {

    companion object {
        @JvmStatic
        @Parameterized.Parameters(name = "{index}: daysBetween({0},{1})={2}")
        fun data(): Iterable<Array<Any>> {
            return arrayListOf(
                arrayOf("21/07/2020", "21/07/2020", 0),
                arrayOf("21/07/2020", "22/07/2020", 1),
                arrayOf("21/07/2020", "23/07/2020", 2),
                arrayOf("21/06/2020", "27/06/2020", 6),
                arrayOf("02/02/2021", "01/02/2021", -1),
                arrayOf("06/02/2021", "02/02/2021", -4),
            )
        }
    }

    @Test
    fun `check days between`() {
        val actual = DateTimeUtils.daysBetween(from = dateFromString.toDate(), to = dateToString.toDate())
        Assert.assertEquals(expected, actual)
    }
}
```

Listing 7.4: Test jednostkowy sprawdzający poprawność działania metody obliczającej długość zakresu dat.

7.7.2. Testy integracyjne

Testy integracyjne należą do testów instrumentacyjnych. Mają one za zadanie przetestować działanie modułów, które zostały zaimplementowane osobno, jednak docelowo przewidywane jest ich współdziałanie.

Z reguły przy niewielkich projektach testy te nie są automatyzowane jak testy jednostkowe, ale przeprowadzane są ręcznie po wprowadzeniu zmian w kodzie celem naprawy błędów, lub po dodaniu nowej funkcjonalności. W tym celu przygotowywane są szczegółowe scenariusze testowe, tak aby łatwo można było odtworzyć taką samą sytuację w przyszłości. Dzięki istnieniu specjalnych bibliotek i pluginów możliwa jest jednak automatyzacja testów integracyjnych poprzez np. symulację interakcji użytkownika z interfejsem. W przypadku aplikacji Planter uciekłam się jednak do testowania manualnego wraz z rozwojem aplikacji.

Istotnym elementem testów integracyjnych jest również sprawdzenie działania aplikacji w zetknięciu z różnymi konfiguracjami sprzętowymi. W tym celu działanie aplikacji Planter zostało przetestowane na trzech różnych urządzeniach:

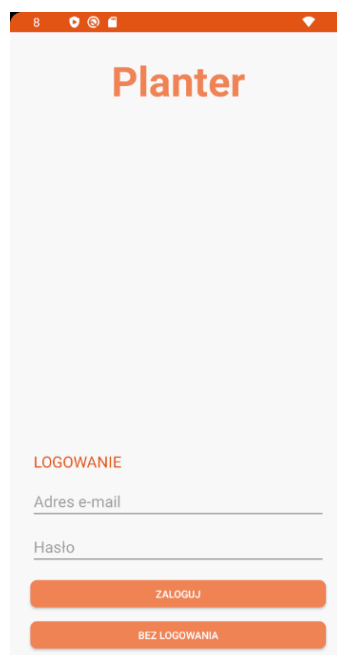
- Pixel 3a, Android 10.0 (smartphone)
- Xiaomi Mi 9Lite, Android 11.0 (smartphone)
- Pixel C, Android 9.0 (tablet)

Niestety poza zasięgiem możliwości znalazło się przetestowanie funkcji aplikacji wymagających połączenia z siecią (rejestracja, logowanie, pobieranie/wysyłanie kopii danych) na tablecie, ze względu na brak dostępu do fizycznego urządzenia oraz do sklepu Play na emulatorach oferowanych przez Android Studio (ze względu na wymagania określone przez Firebase).

7.8. Interfejs użytkownika

7.8.1. Ekran logowania

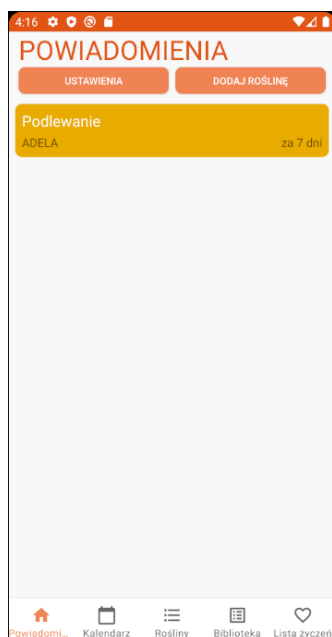
Rysunek 7.4 przedstawia ekran logowania aplikacji Planter.



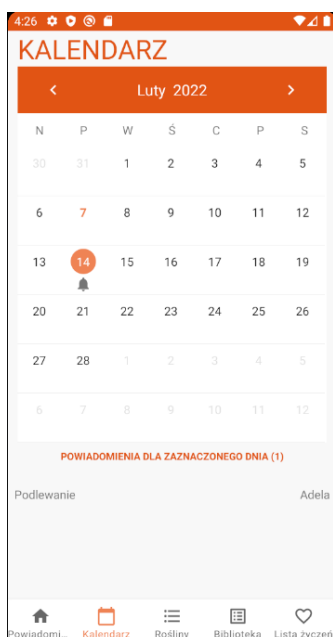
Rysunek 7.4: Ekran logowania. Źródło: opracowanie własne

7.8.2. Ekrany dolnego paska aplikacji

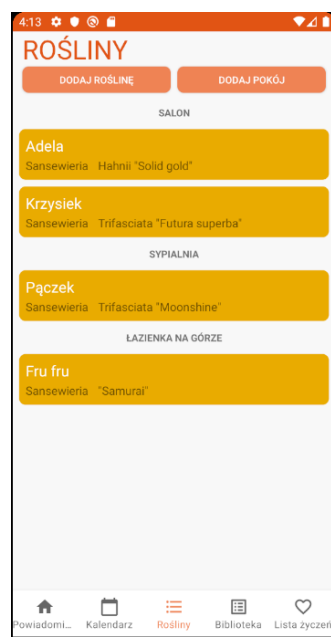
Rysunki 7.5 – 7.9 przedstawiają ekrany dostępne z poziomu dolnego paska aplikacji.



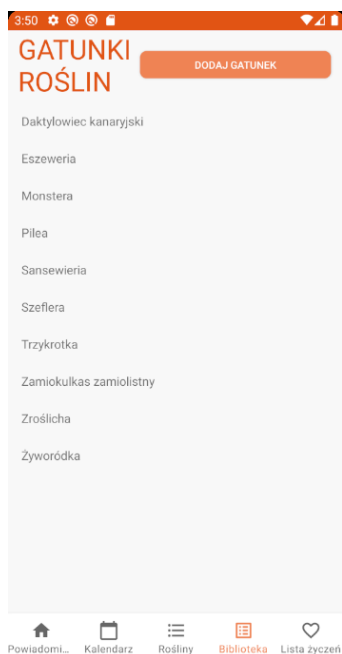
Rysunek 7.5: Ekran aktywnych powiadomień. Źródło: opracowanie własne



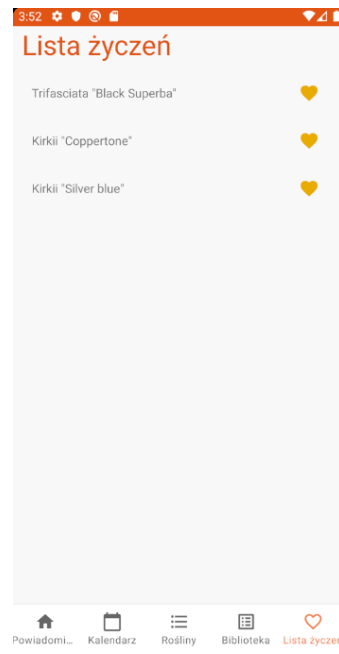
Rysunek 7.6: Ekran widoku kalendarza. Źródło: opracowanie własne



Rysunek 7.7: Ekran widoku listy posiadanych roślin. Źródło: opracowanie własne



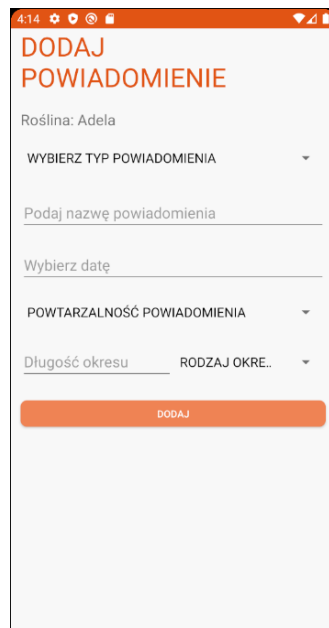
Rysunek 7.8: Ekran widoku biblioteki gatunków roślin. Źródło: opracowanie własne



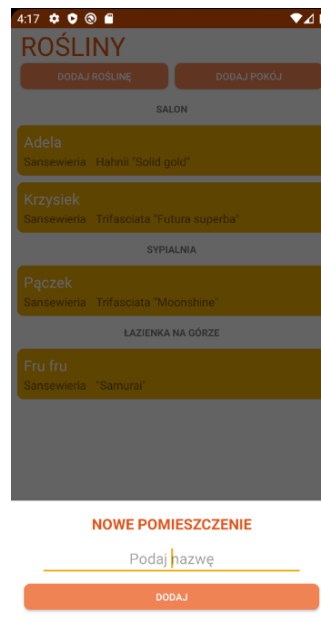
Rysunek 7.9: Ekran widoku listy życzeń. Źródło: opracowanie własne

7.8.3. Ekran dodawania/edycji

Rysunki 7.10 i 7.11 przedstawiają przykładowe ekrany dodawania niektórych rodzajów obiektów.



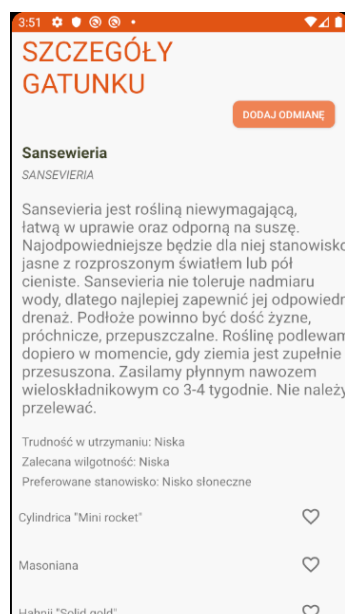
Rysunek 7.10: Ekran dodawania powiadomienia. Źródło: opracowanie własne



Rysunek 7.11: Ekran dodawania pokoju. Źródło: opracowanie własne

7.8.4. Ekran szczegółów obiektów

Rysunki 7.12 i 7.13 przedstawiają ekrany szczegółów gatunku i rośliny.



Rysunek 7.12: Ekran szczegółów gatunku. Źródło: opracowanie własne



Rysunek 7.13: Ekran szczegółów rośliny. Źródło: opracowanie własne

8. Podsumowanie

Celem niniejszej pracy było zaprojektowanie i implementacja aplikacji mobilnej w systemie Android wspomagającej jej użytkowników w zakresie opieki nad ich roślinami doniczkowymi oraz zarządzaniem ich domową kolekcją. W ramach pracy wykonany został przegląd istniejących na rynku rozwiązań, określone technologie, przy użyciu których aplikacja miała zostać wykonana, przedstawiony projekt oraz zaimplementowano i przetestowano aplikację.

W zakres pracy wchodziło zaprojektowanie i wdrożenie mechanizmu przypomnień, które miały ułatwić potencjalnym użytkownikom opiekę nad roślinami, oraz listy życzeń, która z kolei miała za zadanie usprawnić podejmowanie decyzji w zakresie poszerzania kolekcji. Oba te założenia zostały sformułowane na podstawie wykonanej analizy rynkowej i określeniu słabszych stron istniejących rozwiązań. Obydwa te mechanizmy zdejmują z użytkowników konieczność pamiętania pewnych informacji; z jednej strony – kiedy powinni oni wykonać na roślinach zabiegi pielęgnacyjne, z drugiej strony – jakie gatunki i odmiany chcieliby w przyszłości posiadać w swojej kolekcji.

Wybór technologii użytych do implementacji również został podyktowany analizą rynkową. System Android – między innymi ze względu na swoją szeroką dostępność i łatwość obsługi – jest dobrym kandydatem do wytworzenia niniejszej aplikacji. Dodatkowo projekt zakłada korzystanie z licznych bibliotek i rozwiązań oferowanych przez firmę Google. Ich wykorzystanie pozwoliło na usprawnienie procesu implementacji oraz efektywne wykorzystanie możliwości dostarczanych przez system operacyjny.

Założenia opisane w części projektowej zostały osiągnięte, czego dowodem jest powstała implementacja. W ramach analizy technologicznej określono możliwości dalszego rozwoju aplikacji, które zakładają dalsze usprawnienia mechanizmu przypomnień po konsultacji z odpowiednimi ekspertami dziedzinowymi z zakresu opieki nad roślinami, włączenie do aplikacji uczenia maszynowego (które mogłoby dostosowywać się do obecnych warunków temperatury i wilgotności i na tej podstawie, oraz informacji dostarczanych przez użytkownika, szacować odstępy między podlewaniami), oraz rozszerzenie aplikacji o pewne funkcje społecznościowe. Idea opisanych usprawnień w dalszym ciągu sprowadza się do ułatwiania opieki nad roślinami domowymi i poszerzania ich kolekcji.

9. Bibliografia

1. **Richard Goodwin** The History of Mobile Phones From 1973 To 2008: The Cellphones That Made It ALL Happen <https://www.knowyourmobile.com/phones/the-history-of-mobile-phones-from-1973-to-2008-the-handsets-that-made-it-all-happen-d58/> Richard Goodwin [Online]
2. **Pew research center** Mobile fact sheet <https://www.pewresearch.org/internet/fact-sheet/mobile/> [Online]
3. **Statista** Number of smartphone users from 2016 to 2021 (in bilions) <https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/> [Online]
4. **Garden Pals** Houseplant Statistics in 2022 (incl. Covid & Millenials) <https://gardenpals.com/houseplant-statistics/> [Online]
5. **Aaron de Silva** S\$40,000 for a single plant? How much Singaporeans are splurging on their hobby <https://cna.luxury.channelnewsasia.com/obsessions/singapore-plant-collectors-splurging-five-figure-sums-on-hobby-187637> [Online]
6. **John Callaham** The history of Android: The evolution of the biggest mobile OS in the world <https://www.androidauthority.com/history-android-os-name-789433/> [Online]
7. **Statista** Mobile operating systems' market share worldwide from January 2012 to June 2021 <https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009> [Online]
8. **SocialCompare** Android versions comparison <https://socialcompare.com/en/comparison/android-versions-comparison> [Online]
9. **Stack Overflow** 2021 Developer Survey <https://insights.stackoverflow.com/survey/2021> [Online]
10. **Git** Oficjalna strona repozytorium Git <https://git-scm.com/about> [Online]
11. **Java** Oficjalna strona języka Java <https://www.java.com/pl/> [Online]
12. **SHIRSHIR** Android Core: JVM, DVM, ART, JIT, AOT <https://medium.com/programming-lite/android-core-jvm-dvm-art-jit-aot-855039a9a8fa> [Online]
13. **Kotlin** Oficjalna strona języka Kotlin <https://kotlinlang.org/> [Online]
14. **Mariana Berga, Rute Figueiredo** Kotlin vs Java: the 12 differences you should know <https://www.imaginarycloud.com/blog/kotlin-vs-java/> [Online]
15. **Material** Oficjalna strona Material Design <https://material.io> [Online]
16. **Firebase** Oficjalna strona serwisu Firebase <https://firebase.google.com/>
17. **Android** Oficjalna strona dla deweloperów aplikacji w systemie Android <https://developer.android.com/studio/intro> [Online]
18. **Yun Cheng, Aldo Olivares** Advanced Android App Architecture <https://www.raywenderlich.com/books/advanced-android-app-architecture/v1.0/chapters/4-android-architecture-components> [Online]
19. **Android** Oficjalna strona dla deweloperów aplikacji w systemie Android <https://developer.android.com/topic/libraries/architecture/livedata> [Online]
20. **Android** Oficjalna strona dla deweloperów aplikacji w systemie Android <https://developer.android.com/topic/libraries/architecture/viewmodel> [Online]
21. **Deepanshu** Introduction to Room Persistent Library in Android <https://blog.mindorks.com/introduction-to-room-persistent-library-in-android> [Online]
22. **Android** Oficjalna strona dla deweloperów aplikacji w systemie Android <https://developer.android.com/topic/libraries/data-binding> [Online]
23. **Android** Oficjalna strona dla deweloperów aplikacji w systemie Android <https://developer.android.com/guide/navigation> [Online]
24. **Koin** Oficjalna strona frameworka Koin <https://insert-koin.io/> [Online]

25. **Statcounter** Android Version Market Share Worldwide Jan 2021 – Jan 2022
<https://gs.statcounter.com/os-version-market-share/android> [Online]
26. **Alexey Soshin** Hands-on design patterns with Kotlin: Build scalable applications using traditional, reactive, and concurrent design patterns in Kotlin. brak miejsca : Packt Publishing (June 15, 2018). ISBN: 978-1788998017.
27. **Eric Freeman, Elisabeth Robson** Head First Design Patterns: Building Extensible and Maintainable Object-Oriented Software 2nd Edition. Brak miejsca : O'Reilly Media; 2nd edition (December 29, 2020). ISBN: 978-1492078005
28. **Refactoring Guru** Mediator <https://refactoring.guru/pl/design-patterns/mediator> [Online]
29. **Mark Seemann, Steven van Deursen** Dependency Injection Principles, Practices, and Patterns. Brak miejsca : Manning; 1st edition (March 16, 2019). ISBN: 978-1617294730
30. **Andreas Spillner, Tilo Linz** Software Testing Foundations, 5th edition: A Study Guide for the Certified Tester Exam. Brak miejsca : Rocky Nook; 5th edition (July 21, 2021). ISBN: 1681988534
31. **Vladimir Khorikov** Unit Testing Principles, Practices, and Patterns: Effective testing styles, patterns, and reliable automation for unit testing, mocking and integration testing with examples in C#. brak miejsca : Manning; 1st edition (January 14, 2020). ISBN: 9781617296277
32. **LemonClip** Garden Manager
<https://play.google.com/store/apps/details?id=com.jee.green&hl=pl&gl=US> [Online]
33. **Christian Castaldi** Planteo : Plant diary & reminder
<https://play.google.com/store/apps/details?id=com.kriskast.planteo> [Online]
34. **Bloomscape** Vera: Plant Care Made Simple
<https://play.google.com/store/apps/details?id=us.crema.vera> [Online]
35. **Nikola Kosev** Waterbot: Podlewanie roślin
<https://play.google.com/store/apps/details?id=net.kosev.watering> [Online]
36. **Aleksei Turbin** Plant Care Reminder – Plant Watering
<https://play.google.com/store/apps/details?id=com.atcorapps.plantcarereminder> [Online]

9.1. Spis rysunków

Rysunek 2.1 Wykres liczby użytkowników telefonów typu smartphone na świecie w latach 2016-2026 (prognozowane). Źródło: [3].....	6
Rysunek 4.1: Aplikacja Waterbot - dodawanie nowej rośliny. Źródło: [35].....	9
Rysunek 4.2: Aplikacja Planteo - ustawienia powiadomień dla rośliny. Źródło: [33].....	9
Rysunek 4.3: Aplikacja Plant Care Reminder - dodawanie nowej rośliny. Źródło: [36].....	10
Rysunek 4.4: Aplikacja Garden Manager - dodawanie nowej rośliny Źródło: [32].....	10
Rysunek 4.6: Aplikacja Vera - lista zadań do wykonania. Źródło: [34].....	11
Rysunek 2.7: Aplikacja Vera - lista roślin. Źródło: [34].....	11
Rysunek 4.8: Aplikacja Vera - dodawanie rośliny, harmonogram. Źródło: [34].....	11
Rysunek 5.1: Wykres przedstawiający procentowy udział w rynku poszczególnych systemów operacyjnych na urządzenia mobilne w latach 2012-2021. Źródło: [7].....	13
Rysunek 5.2: Wykres ilustrujący popularność narzędzia Git wśród użytkowników portalu StackOverflow. Źródło: [9].....	15
Rysunek 6.1: Diagram przypadków użycia aplikacji Planter. Źródło: opracowanie własne.....	22
Rysunek 6.2: Schemat bazy danych aplikacji Planter. Źródło: opracowanie własne.....	38
Rysunek 6.3: Diagram klas pakietu data. Źródło: opracowanie własne.....	40
Rysunek 6.4: Diagram klas pakietu domain. Źródło: opracowanie własne.....	41
Rysunek 6.5: Diagram klas pakietu presentation. Źródło: opracowanie własne.....	42
Rysunek 6.6: Projekt ekranu listy aktywnych powiadomień. Źródło: opracowanie własne....	43

Rysunek 6.7: Projekt ekranu widoku kalendarza. Źródło: opracowanie własne.....	43
Rysunek 6.8: Projekt ekranu widoku posiadanych roślin. Źródło: opracowanie własne.....	43
Rysunek 6.9: Projekt ekranu widoku biblioteki gatunków roślin. Źródło: opracowanie własne.....	44
Rysunek 6.10: Projekt ekranu widoku listy życzeń. Źródło: opracowanie własne.....	44
Rysunek 7.1: Diagram pakietów, część pierwsza. Źródło: opracowanie własne.....	45
Rysunek 7.2: Diagram pakietów, część druga. Źródło: opracowanie własne.....	45
Rysunek 7.3: Diagram pakietów, część trzecia. Źródło: opracowanie własne.....	45
Rysunek 7.4: Ekran logowania. Źródło: opracowanie własne.....	51
Rysunek 7.5: Ekran aktywnych powiadomień. Źródło: opracowanie własne.....	52
Rysunek 7.6: Ekran widoku kalendarza. Źródło: opracowanie własne.....	52
Rysunek 7.7: Ekran widoku listy posiadanych roślin. Źródło: opracowanie własne.....	52
Rysunek 7.8: Ekran widoku biblioteki gatunków roślin. Źródło: opracowanie własne.....	52
Rysunek 7.9: Ekran widoku listy życzeń. Źródło: opracowanie własne.....	52
Rysunek 7.10: Ekran dodawania powiadomienia. Źródło: opracowanie własne.....	53
Rysunek 7.11: Ekran dodawania pokoju. Źródło: opracowanie własne.....	53
Rysunek 7.12: Ekran szczegółów gatunku. Źródło: opracowanie własne.....	54
Rysunek 7.13: Ekran szczegółów rośliny. Źródło: opracowanie własne.....	54

9.2. Spis tabel

Tabela 5.1: Udział rynkowy poszczególnych wersji systemu Android. Stan – Styczeń 2022. Źródło: [25].....	14
Tabela 5.2: Podsumowanie bibliotek zewnętrznych, użytych w trakcie implementacji. Źródło: opracowanie własne.....	19
Tabela 6.1: PU#001 Wyświetlenie listy aktywnych przypomnień.....	23
Tabela 6.2: PU#002 Utworzenie nowego przypomnienia.....	23
Tabela 6.3: PU#003 Usunięcie aktywnego przypomnienia.....	24
Tabela 6.4: PU#004 Oznaczenie przypomnienia jako wykonane.....	24
Tabela 6.5: PU#005 Dodanie nowego pokoju.....	25
Tabela 6.6: PU#006 Usunięcie istniejącego pokoju.....	25
Tabela 6.7: PU#007 Wyświetlenie kalendarza przypomnień.....	26
Tabela 6.8: PU#008 Wyświetlenie biblioteki gatunków i odmian roślin.....	26
Tabela 6.9: PU#009 Dodanie nowego gatunku do biblioteki.....	27
Tabela 6.10: PU#010 Edycja dodanego przez użytkownika gatunku.....	28
Tabela 6.11: PU#011 Usunięcie dodanego przez użytkownika gatunku.....	28
Tabela 6.12: Pu#012 Wyświetlenie szczegółów gatunku roślin.....	29
Tabela 6.13: PU#013 Dodanie nowej odmiany do istniejącego gatunku roślin.....	29
Tabela 6.14: PU#014 Edycja dodanej przez użytkownika odmiany roślin.....	30
Tabela 6.15: PU#015 Usunięcie dodanej przez użytkownika odmiany roślin.....	30
Tabela 6.16: PU#016 Wyświetlenie listy posiadanych roślin.....	31
Tabela 6.17: PU#017 Dodanie nowej rośliny do listy posiadanych.....	31
Tabela 6.18: PU#018 Edycja istniejącej rośliny.....	32
Tabela 6.19: PU#019 Usunięcie istniejącej rośliny.....	32
Tabela 6.20: PU#020 Wyświetlenie danych rośliny.....	33
Tabela 6.21: PU#021 Wyświetlenie ustawień aplikacji.....	33
Tabela 6.22: Pu#022 Założenie konta użytkownika.....	33
Tabela 6.23: PU#023 Zalogowanie.....	34
Tabela 6.24: PU#024 Zapis kopii danych na serwerze.....	34

Tabela 6.25: PU#025 Przywracanie danych z kopii.....	35
Tabela 6.26: PU#026 Wylogowanie.....	35
Tabela 6.27: PU#027 Wyświetlenie listy życzeń.....	36
Tabela 6.28: PU#028 Dodanie odmiany roślin do listy życzeń.....	36
Tabela 6.29: PU#029 Usunięcie odmiany roślin z listy życzeń.....	37
Tabela 7.1: Pakiety aplikacji Planter.....	46
Tabela 7.2: PI#001 Wyświetlenie przypomnień ustawionych przez użytkownika w formie kalendarza.....	49

9.3. Spis listingów

Listing 7.1: Kod w języku Kotlin inicjujący bazę danych.....	48
Listing 7.2: Kod w języku Kotlin definiujący encję bazy danych (przykład).....	48
Listing 7.3 Kod w języku Kotlin definiujący metody obiektu dostępu do danych (przykład)	49
Listing 7.4: Test jednostkowy sprawdzający poprawność działania metody obliczającej długość zakresu dat.....	50