

WYDZIAŁ PODSTAWOWYCH PROBLEMÓW TECHNIKI
POLITECHNIKA WROCŁAWSKA

SYSTEM DO ZARZĄDZANIA BIZNESEM

KRZYSZTOF SZAFRANIAK
NR INDEKSU: 244932

Praca inżynierska napisana
pod kierunkiem
dr inż. Marcin Zawada



Politechnika
Wrocławskie

WROCŁAW 2020

Spis treści

1 Wstęp	1
2 Wprowadzenie	3
2.1 System Android	3
2.2 JSON Web Token	4
2.3 Interfejs programistyczny aplikacji REST	4
3 Analiza problemu	5
3.1 Założenia funkcjonalne	5
3.2 Założenia pozafunkcjonalne	5
3.3 Graficzny interfejs użytkownika	6
3.4 Udział poszczególnych wersji systemu Android na rynku	6
3.5 Uwierzytelnienie i autoryzacja użytkownika	7
3.6 Uwierzytelnienie z wykorzystaniem zewnętrznych usług	9
4 Projekt systemu	11
4.1 Model biznesowy	11
4.2 Diagram komponentów	12
4.3 Baza danych	12
5 Implementacja systemu	17
5.1 Środowisko programisty	17
5.2 Mechanizm filtrowania treści	17
5.3 Analiza problemów implementacyjnych	18
5.4 Biblioteki i narzędzia	20
5.5 Wykorzystane wzorce projektowe	22
6 Interfejs użytkownika	23
6.1 Widok ekranu logowania	23
6.2 Widoki główne	24
6.3 Widok kontaktów	25
6.4 Widok zasobów i produktów w magazynie	26
6.5 Widok modeli produktów i usług	27
6.6 Widok listy faktur i ich szczegółów	28
6.7 Widok zdarzeń finansowych	29
6.8 Generowanie faktur	30
6.9 Widok ustawień i praw autorskich	32
6.10 Nietekstowe metody wprowadzania danych	33
7 Konfiguracja i uruchomienie systemu	35
7.1 Środowisko	35
7.2 Konfiguracja usług zewnętrznych	35
7.3 Konfiguracja systemu	37
7.4 Instalacja i uruchomienie	38

8 Podsumowanie	39
Bibliografia	41
A Zawartość płyty CD	43

Wstęp

Przedmiotem pracy jest stworzenie systemu wspierającego przedsiębiorców w prowadzeniu niewielkich działalności gospodarczych. Najbardziej zaawansowane systemy biznesowe stosowane w korporacjach i rozwijane przez dziesiątki specjalistów, biorą aktywny udział we wszystkich sektorach funkcjonowania firm. Pozwalają modelować oferty, kontrolują sprzedaż, odpowiadają za dostarczanie usług, podtrzymywanie relacji z klientami, marketing, księgowość i wiele innych. Stosowanie rozwiązań tego typu - mniej lub bardziej zaawansowanych - pozwala efektywniej wykorzystać potencjał przedsiębiorstwa i usprawnić zachodzące w nim procesy biznesowe. W zrealizowanym w ramach pracy projekcie, skupiono się na wsparciu jedno- lub maksymalnie kilkuosobowych działalności.

Celem było zaprojektowanie i napisanie systemu informatycznego, który będzie dostosowany do potrzeb przedsiębiorców. Zarówno dla firm, które opierają się na sprzedaży produktów, jak też tych oferujących usługi. Istotnym elementem oprogramowania jest możliwość fakturowania. Informatyzacja tego procesu usprawni funkcjonowanie działalności, a dodatkowa funkcja monitorowania statusów płatności dla wystawionych dokumentów, pozwoli zadbać o to, aby przedsiębiorca nie przeoczył żadnej z wynikającej z nich należności. Powiązana z tym możliwość kolekcjonowania dokumentów sprawia, że użytkownik osiąga korzyść z tytułu zapisania ich w jednym miejscu. Zoptymalizowanie procesu generowania faktur poprzez wykorzystanie zapisanych w systemie modeli produktów i usług, pozwoli zaoszczędzić czas na uzupełnianie formularzy.

Ważnym elementem systemu biznesowego jest funkcjonalność prowadzenia rejestru stanu produktów w magazynach. Pozwala ona w każdym momencie zweryfikować, czy dany zasób znajduje się w magazynie. Dla tych z przedsiębiorców, którzy będą weryfikować bilans przychodów oraz rozchodów ważną składową systemu jest możliwość prowadzenia ewidencji zdarzeń finansowych.

Ze względu na różnice wynikające ze specyfiki działania firm, oprogramowanie powinno być dostępne na urządzenia mobilne. Przedsiębiorcy wykonujący pracę u klienta osiągną korzyści z rozwiązania, które będzie im towarzyszyć na kompaktowym urządzeniu, poza siedzibą firmy. Omówione wcześniej funkcjonalności fakturowania i monitorowania stanu magazynowego, umożliwiają takim przedsiębiorcom (np. zajmującym się serwisowaniem urządzeń) zweryfikować posiadanie potrzebnego zasobu bez względu na miejsce, w którym się znajdują. Po zakończeniu wykonywania usługi lub sprzedaży daje możliwość bezzwłocznego wygenerowania faktyry, stwarzając korzyści dla obu stron.

Szczególnie istotnym elementem oprogramowania biznesowego jest intuicyjny i funkcjonalny interfejs użytkownika, który towarzyszy mu przez cały okres korzystania z systemu. Spójny zarówno funkcjonalnie i graficznie interfejs, usprawnia proces, w którym użytkownik uczy się nim posługiwać. Często jest najważniejszym elementem systemu, ponieważ umożliwia interakcję z użytkownikiem, a jego projekt i wykonanie bezpośrednio wpływa na postrzeganie całego oprogramowania.

Zakres pracy obejmuje następujące zadania:

- implementacja serwera REST API udostępniającego interfejs dla aplikacji mobilnej,
- przeprowadzenie konfiguracji oprogramowania uwierzytelniającego Keycloak oraz jego integracja z serwerem REST,
- zaprojektowanie i wykonanie ergonomicznego interfejsu użytkownika,
- implementacja aplikacji na system Android,
- zaprojektowanie i zbudowanie bazy danych MySQL,
- wykonanie konfiguracji usług dostarczanych przez firmy Google i Facebook oraz ich integracja z oprogramowaniem Keycloak i aplikacją mobilną,



- wdrożenie komponentu, który pozwoli odczytać kod kreskowy przy użyciu kamery wbudowanej w urządzenie z systemem Android,
- opracowanie rozwiązania problemu generowania faktur w formacie PDF.

Praca składa się z ósmu rozdziałów. Rozdział 2 („[Wprowadzenie](#)”) w sposób zwięzły przedstawia podłożę pracy, zmiany jakie dokonują się na naszych oczach w sposobie korzystania z urządzeń elektronicznych. Przybliża również bazowe terminy takie jak REST API, JWT, czy Android stanowiące podstawę do realizacji systemu.

Rozdział 3 („[Analiza problemu](#)”) przeprowadza analizę tematyki, której poświęcona jest niniejsza praca. Opisuje wymagania funkcjonalne i pozafunkcjonalne. Przedstawia opracowany mechanizm uwierzytelnienia i autoryzacji użytkowników, a także punktuje zasady, którymi należy się kierować projektując graficzny interfejs użytkownika. Zawiera analizę udziału poszczególnych wersji systemu Android na rynku i przedstawia proces myślowy uzasadniający wybór tych z nich, które będą wspierane.

Rozdział 4 („[Projekt systemu](#)”) omawia przyjęty model biznesowy. Przy użyciu diagramu pokazuje z jakich komponentów jest zbudowany system oraz w jaki sposób są one ze sobą powiązane. Zawiera projekt bazy danych, na który składa się jej schemat, opis tabel oraz rozwiązań, wynikających z analizy wymagań systemu.

Rozdział 5 („[Implementacja systemu](#)”) prezentuje środowisko programistyczne oraz nowoczesne rozwiązania, wykorzystane do realizacji systemu. Przedstawia sposób implementacji wybranych mechanizmów, zawiera opis stosowanych wzorców projektowych oraz zewnętrznych bibliotek. Przeprowadza analizę problemów, które wystąpiły podczas rozwoju oprogramowania.

Rozdział 6 („[Interfejs użytkownika](#)”) jest poświęcony interakcji aplikacji mobilnej z użytkownikiem. W zwięzły sposób opisuje znaczenie dobrze zaprojektowanego interfejsu. Przedstawia dostępne dla użytkownika widoki, analizuje ich budowę oraz opisuje zachodzące na nich procesy.

Rozdział 7 („[Konfiguracja i uruchomienie systemu](#)”) opisuje kroki, które należy wykonać, aby uruchomić system. Większa część treści jest poświęcona przeprowadzeniu konfiguracji zewnętrznych usług, które dotyczą prywatnych danych i wymagają indywidualnych ustawień.

Rozdział 8 („[Podsumowanie](#)”) jest podsumowaniem zrealizowanej pracy i zaimplementowanego w jej ramach systemu. Weryfikuje, czy ustalone wymagania zostały zrealizowane, a także zawiera rozważanie na temat możliwości dalszego rozwoju oprogramowania.

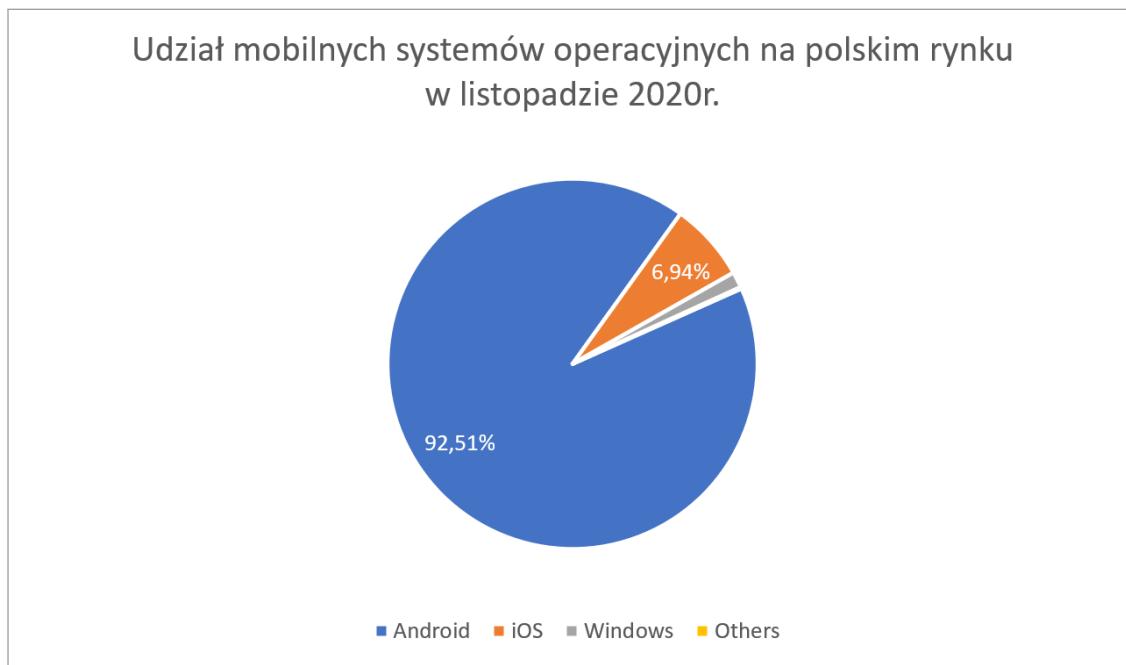
Wprowadzenie

Szybko rozwijająca się technologia oraz powstanie wydajnych i przenośnych urządzeń spowodowało, że obecnie obserwowany jest trend, w którym odchodzi się od rozwiązań stacjonarnych, a coraz częściej korzysta się z urządzeń przenośnych. Nowoczesne smartfony tylko w niewielkim stopniu przypominają jeszcze swoich poprzedników, czyli klasyczne telefony komórkowe. Ich obecna forma odziedziczyła nie tylko funkcjonalności starszych urządzeń, ale również rozszerzyła je o wiele nowych. Dziś, smartfony dają możliwości, które kiedyś było można osiągnąć jedynie z wykorzystaniem kilku lub kilkunastu osobnych urządzeń. Szeroka gama zastosowań i kompaktowy format urządzenia sprawiają, że smartfony stały się niezwykle popularne.

W wyniku tych zmian rośnie zainteresowanie klientów systemami rozwijanymi na platformy mobilne. Wśród nich są również jednoosobowe i małe firmy poszukujące rozwiązań dla siebie, dzięki którym z poziomu smartfonu lub tabletu będą mogły obserwować stan oraz zarządzać podstawowymi składowymi swoich działalności.

W tym rozdziale omówiono kluczowe zagadnienia, które są podstawą implementowanego oprogramowania. Opisano udział mobilnych systemów operacyjnych na polskim rynku oraz platformę Android, na którą tworzona jest aplikacja mobilna. Poza tym, przybliża hasła na których opiera się komunikacja sieciowa w systemie.

2.1 System Android



Rysunek 2.1: Udział mobilnych systemów operacyjnych na polskim rynku w listopadzie 2020r. [17]



Smartfony oraz tablety regularnie przejmują zadania przeznaczone dla komputerów stacjonarnych i laptopów. Według danych na listopad 2020r. opublikowanych przez Global Stats dominującą część tych urządzeń jest wyposażona w jeden z dwóch popularnych systemów (Rysunek 2.1), iOS rozwijany przez firmę Apple oraz Android, którego producentem jest konsorcjum Open Handset Alliance[17]. Ich udział w rynku rozkłada się odpowiednio 6.64% dla pierwszego z nich oraz 92.82% dla systemu Android. Niniejszej pracy skupia się na drugim z nich.

Android jest systemem uniwersalnym[1]. Bez znaczenia na markę i konkretny model urządzenia, na którym zostanie zainstalowany, udostępnia użytkownikowi jednolity interfejs. Oprogramowanie wykorzystuje jądro Linux, a jego kod źródłowy jest publicznie dostępny[9]. Otwartość systemu powoduje, że producenci smartfonów i tabletów często wprowadzają autorskie modyfikacje. Nakładki systemowe pozwalają markom, wyróżnić się na tle konkurencji. Wprowadzenie dodatkowych cech, na przykład zmiana zachowania i wyglądu interfejsu użytkownika pozwala budować jej własną tożsamość. Powstaje również wiele niekomercyjnych wersji systemu Android, oferują one funkcje niedostępne w oprogramowaniu dołączanym do urządzeń. Często wgranie takiej wersji jest jedyną drogą pozyskania aktualizacji systemu dla użytkowników, którzy korzystają z starszego urządzenia, a ich producent zaprzestał rozwijania i publikowania aktualizacji.

Popularność tego systemu i jego obecność na urządzeniach praktycznie każdej marki z wyjątkiem Apple, zachęca programistów do tworzenia aplikacji, których obecnie jest już ponad 3 miliony[18]. Do implementacji systemów na tej platformie, twórcy mogą wykorzystać popularny język Java. Dzięki temu, dużo łatwiej jest im rozpoczęć pracę nad aplikacją na system Android.

2.2 JSON Web Token

JWT to otwarty standard, który umożliwia wymianę informacji. Nazwa formatu wykorzystawanego do zapisu danych zawiera się w nazwie standardu. Jest nim bardzo popularny i czytelny dla człowieka format JSON. Transmisja danych jest zakodowana z wykorzystaniem jednego z dostępnych algorytmów szyfrowania. Celem nie jest ukrycie danych, ale uzyskanie pewności, że zostały wydane przez wiarygodne źródło. Najczęściej jest wykorzystywany do autoryzacji dostępu.

JSON Web Token składa się z trzech części oddzielonych od siebie kropkami. Pierwszy fragment to nagłówek. Najczęściej składa się z dwóch pól: informacji o typie klucza oraz wybranym algorytmie szyfrowania. Fragment ten pozwala poprawnie dobrze metodę pozwalającą odkodować klucz. Środkowy element reprezentuje ładunek/zawartość. W tej sekcji umieszcza się dane przeznaczone do przesłania. Oprócz danych identyfikujących, często dołącza się również informację o terminie ważności klucza czy roli użytkownika. Trzecim elementem tworzącym JWT jest sygnatura. Jest to podpis cyfrowy potwierdzający autentyczność danych. Dzięki niemu możliwe jest potwierdzenie, że nadawca jest tym za kogo się podaje oraz że otrzymane dane są w niezmienionej formie. Każda z wymienionych części zostaje przedstawiona przy użyciu kodu Base64 w celu zmiany struktury informacji, a następnie połączona w jeden ciąg znaków oddzielony dwoma kropkami.

2.3 Interfejs programistyczny aplikacji REST

Representational State Transfer Application Programming Interface to uniwersalny interfejs sieciowy oparty na ustalonych zasadach i wykorzystywany do komunikacji pomiędzy aplikacją klienta, a serwerem. Transmisja danych odbywa się za pośrednictwem protokołu HTTP. Udostępnia on dziewięć metod, z których najczęściej wykorzystuję się cztery pozwalające na odczyt, zapis, aktualizację oraz usuwanie danych. Są nimi odpowiednio GET, POST, PUT, DELETE. Odpowiedź serwera składa się z kodu oraz danych, które zazwyczaj są przesyłane w formacie JSON, rzadziej XML. Ich uniwersalność i przyjazna forma ułatwia proces testowania interfejsów sieciowych.

Aby interfejs sieciowy można było określić mianem REST musi on być bezstanowy. Oznacza to, że każde żądanie klienta musi zawierać komplet informacji oraz że serwer nie przechowuje danych o sesji użytkownika. Istotnym aspektem jest odseparowanie interfejsu użytkownika od serwera przechowującego informacje. Zaletą takiego rozwiązania jest duża mobilność między platformowa. Pozwala ono na tworzenie wielu niezależnych od siebie aplikacji zintegrowanych z interfejsem przy jednoczesnym zachowaniu spójności danych.

Analiza problemu

W tym rozdziale przedstawiono analizę zagadnienia, które podlega informatyzacji. Opracowano i opisano założenia funkcjonalne i pozafunkcjonalne. Przedstawiono procesy uwierzytelnienia i autoryzacji, a także zasady, którymi należy się kierować projektując graficzny interfejs użytkownika. Dokonano analizy udziału poszczególnych wersji systemu Android i uzasadniono wybór tych z nich, które będą wspierane.

3.1 Założenia funkcjonalne

Oczekiwania dotyczące zachowania systemu oraz problemy jakie rozwiązuje zostały opisane w postaci wymagań funkcjonalnych.

- Oprogramowanie pozwala użytkownikowi wystawiać faktury, będące zgodnymi z polskim prawem podatkowym.
- Modele produktów i usług są przechowywane w systemie.
- Interfejs użytkownika jest dostępny w języku polskim i angielskim.
- System udostępnia funkcję monitorowania stanu i modyfikacji spisu zasobów w magazynach.
- W celu odnalezienia interesującego użytkownika towaru w systemie, użytkownik ma możliwość zeskanowania jego kodu kreskowego.
- Aplikacja przechowuje kontakty związane z prowadzonym biznesem.
- Gdy to możliwe, formularze znajdujące się w aplikacji mobilnej pozwalają skorzystać z danych zapisanych w systemie, poprzez wybór jednej z sugerowanych podpowiedzi.
- Użytkownik może prowadzić rejestr przychodów i rozchodów oraz aktualizować stan opłacenia faktur.
- Proces logowania i rejestracji przebiega z wykorzystaniem konta utworzonego w portalach Google lub Facebook oraz użytkownik ma możliwość uruchomienia opcji automatycznego logowania do systemu z poziomu aplikacji mobilnej.

3.2 Założenia pozafunkcjonalne

Pożądane cechy i wymagania jakościowe, które mają znaczący wpływ na wybór architektury i rozwiązania niewidoczne dla użytkownika systemu zostały przedstawione za pomocą wymagań niefunkcjonalnych.

- Ergonomiczny i funkcjonalny interfejs użytkownika.
- Aplikacja mobilna wspiera urządzenia wyposażone w system Android od wersji 8.0.
- Oprogramowanie serwerowe jest skonteneryzowane oraz może zostać uruchomione na dowolnej maszynie wyposażonej w aplikację Docker^[3].
- Wszystkie dane użytkownika znajdują się na zdalnym serwerze i umożliwia dostęp do informacji z dolnej instancji aplikacji mobilnej. Komunikacja odbywa się przy użyciu REST API,



3.3 Graficzny interfejs użytkownika

Interfejs systemu to fragment oprogramowania, który umożliwia komunikację systemu z użytkownikiem. Wyróżnia się ich wiele rodzajów, jednak obecnie najpopularniejszym rozwiązaniem jest projektowanie interfejsów graficznych. Jest to typ, w którym sposób prezentacji informacji - poza tekstem - odbywa się za pomocą rysunków, animacji, obrazów i widżetów. Taka forma trafia do większej ilości użytkowników i jest łatwiejsza w odbiorze. W dalszej części zostaną opisane cechy i zasady, które określają dobrze zaprojektowany interfejs graficzny.

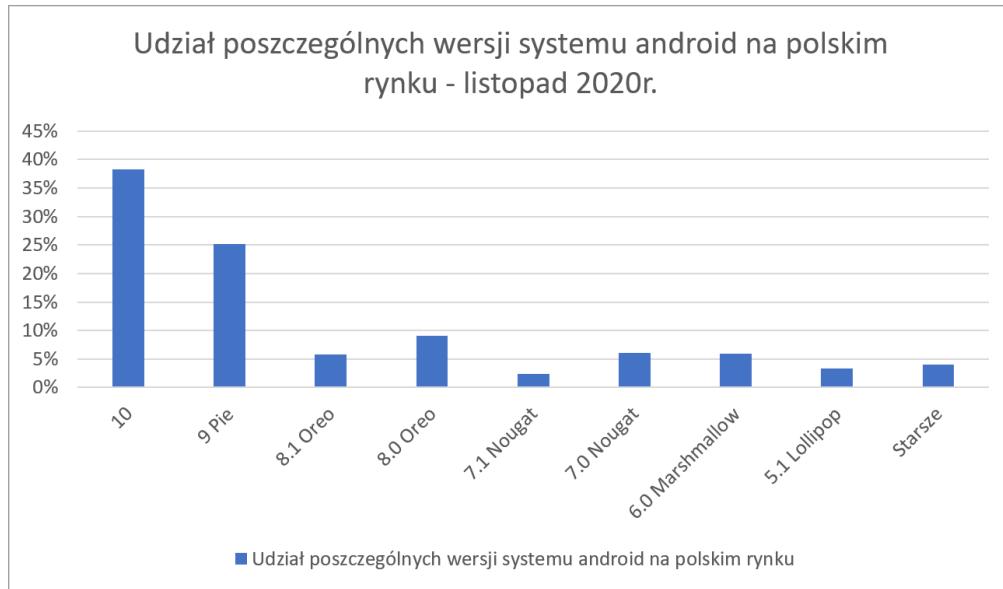
Interfejs powinien być intuicyjny. Miarą tej cechy jest łatwość, z jaką użytkownik się po nim porusza przy pierwszym zetknięciu z systemem. Powinien on bez trudu dostrzegać jego funkcjonalne elementy i właściwie je rozumieć. Akcje ukryte pod ikonami powinny wiązać się z tym co prezentują i być dostosowane do ogólnie panujących norm i przyzwyczajeń klientów.

Istotną cechą dobrze zaprojektowanego interfejsu jest spójność. Określa ona konsekwentne realizowanie układów graficznych dla wszystkich widoków. Należy zadbać, aby komponenty realizujące podobne funkcjonalności na różnych ekranach znajdowały się w tych samych miejscach i wyglądały podobnie. Widoki powinny korzystać z wspólnego motywów i sprawiać wrażenie całości.

Projekt powinien mieć możliwe prostą strukturę, aby użytkownik nie zabłądził przełączając się pomiędzy ekranami. Należy unikać rozwiązań, w których bez końca możliwe jest przełączanie się w głąb interfejsu. Dobrą praktyką jest ograniczenie struktury do kilku poziomów oraz informowanie użytkownika o aktualnej pozycji w systemie. Funkcjonalność definiuje przejrzystość i przemyślane rozmieszczenie funkcji. System powinien komunikować błędy występujące podczas działania systemu oraz zapewniać użytkownikom pomoc. Należy kontrolować wprowadzane dane pod kątem poprawności i informować o nieprawidłowościach.

Projektując komponenty należy dobrać odpowiednie kształty i kolory, które skojarzeniami nawiązują do pełnionej funkcji. Najbardziej charakterystycznymi kolorami wykorzystywanymi w interfejsach są czerwony, żółty i zielony. Pierwszy kojarzony z błędem lub przerwaniem operacji, drugi to kolor ostrzeżeń, a ostatni - zielony - odnosi się do sukcesu, akceptacji lub rozpoczęcia procesu. Dobrą praktyką jest unikanie jaskrawych kolorów, które mogą obciążać wzrok użytkownika.

3.4 Udział poszczególnych wersji systemu Android na rynku



Rysunek 3.1: Udział poszczególnych wersji systemu Android na polskim rynku – listopad 2020r. [18]

Otwartość systemu Android, która została opisana we wprowadzeniu (Rozdział 2.1), powoduje również niekorzystne zjawiska. Istnienie wielu wersji oprogramowania sprawia, że producent nie ma możliwości dostarczenia aktualizacji, pasującej do wszystkich urządzeń. Skutkuje to dużą fragmentacją mobilnego systemu operacyjnego i brakiem lub opóźnieniem publikacji jego nowej wersji dla zmodyfikowanych systemów. Projektując aplikację oczekuje się między innymi jej dostępności na możliwie największej grupie urządzeń oraz wykorzystania najnowszych rozwiązań technologicznych. Fragmentacja systemu na poszczególne wersje sprawia, że rozwiązania dotyczące wymienionych aspektów zachodzą w konflikt, a zadaniem projektanta jest znaleźć kompromis, który będzie optymalnym rozwiązaniem.

Analiza udziału poszczególnych wersji systemu Android na polskim rynku (Rysunek 3.1) wskazuje, że Polacy w zdecydowanej większości korzystają z nowych wydań. Niestety na rynku wciąż znajdują się starsze urządzenia, którym najczęściej towarzyszy starsze oprogramowanie. Należy pamiętać, że aplikacja wykorzystująca nowe funkcje wprowadzone w danej wersji systemu, nie będzie kompatybilna z urządzeniami z starszym systemem. Jednak nowe wersje systemu wprowadzają nowe rozwiązania i możliwości, które mogą być wykorzystane przez programistów.

Realizowane rozwiązanie będzie wspierać system od wersji 8.0, co pozwoli na zainstalowanie aplikacji na 78,33% z wszystkich urządzeń z systemem Android na polskim rynku. Taka decyzja pozwala wykorzystać zestaw narzędzi SDK (Software Development Kit Platform-tools) w wydaniu 26, do którego między innymi włączono wsparcie dla języka Java 8. Ta wersja odznacza się przede wszystkim wprowadzeniem do składni wyrażeń lambda oraz referencji metod. Decyzja jest więc kompromisem, który pozwala wykorzystać nowoczesne rozwiązania i jednocześnie umożliwić instalację aplikacji na jak największej liczbie urządzeń.

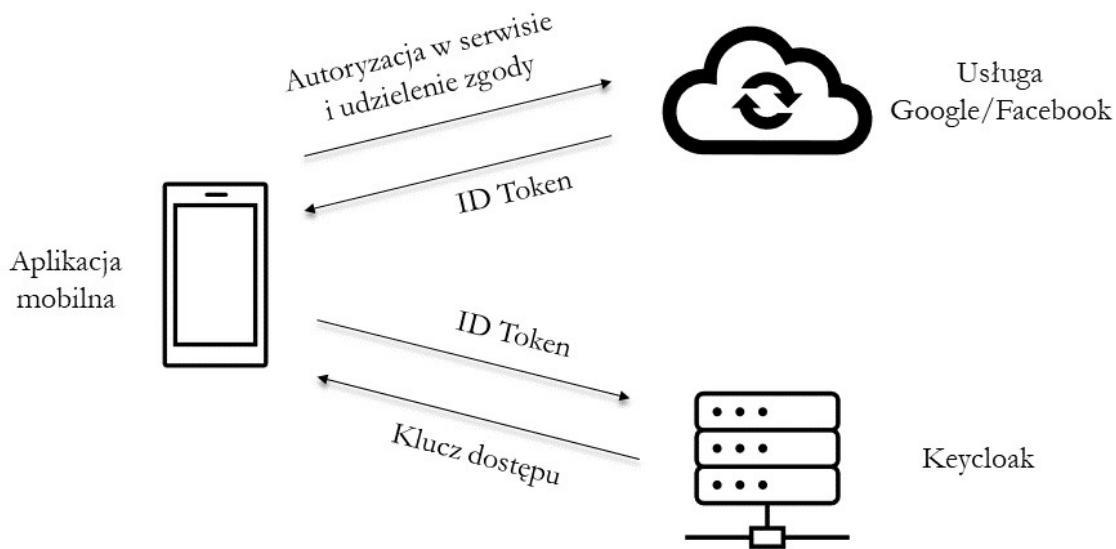
3.5 Uwierzytelnienie i autoryzacja użytkownika

Autoryzacja to proces polegający na potwierdzeniu czy użytkownik ma prawo dostępu do określonych zasobów bądź wykonania operacji. Celem tego procesu jest kontrola dostępu. Do jej przeprowadzenia konieczne jest wcześniejsze ustalenie tożsamości użytkownika, co nazywane jest uwierzytelnieniem.

Keycloak jest otwartym oprogramowaniem zarządzającym uwierzytelnieniem i autoryzacją użytkowników [8]. Oprogramowanie może być łączone z wieloma systemami jednocześnie, a jego konfiguracja pozwala na logowanie z użyciem kont założonych w innych serwisach. Producenci oprogramowania dostarczają adaptery do aplikacji klienckich na różne platformy, które ułatwiają ich integrację. Keycloak udostępnia również rozbudowany interfejs graficzny w postaci panelu administracyjnego dostępnego za pośrednictwem przeglądarek, jednak nie zawsze jest on wystarczający. Pozwala zarządzać zarejestrowanymi użytkownikami, integracją systemów oraz procesami związanymi z autoryzacją i uwierzytelnieniem. Keycloak posiada własną bazę użytkowników. Jego główną funkcją jest ich uwierzytelnienie w systemach go wykorzystujących. Umożliwia również kontrolę dostępu do powiązanych z nim systemów, więc jest również systemem autoryzacji, jednak ta funkcjonalność jest wykorzystywana najczęściej w przypadku większej liczby powiązanych systemów. Oprogramowanie realizowane w niniejszej pracy wymaga dodatkowego mechanizmu autoryzacji, który ograniczy dostęp uwierzytelnionych użytkowników tylko do tych zasobów, których jest właścicielem.

Uwierzytelnienie użytkownika składa się z dwóch etapów. Celem pierwszego z nich jest uzyskanie od usługi zewnętrznej klucza identyfikującego (ID Token). Jest to identyfikator JWT zawierający informacje pozwalające zweryfikować użytkownika m.in. jego unikalny numer. Aby go uzyskać klient musi zalogować się na konto Google/Facebook, a następnie wyrazić zgodę na przekazanie informacji do aplikacji, która ma na celu wykorzystanie ich do uwierzytelnienia użytkownika. Drugi etap odbywa się w tle i nie wymaga interakcji z aplikacją. Wykonywane jest zapytanie do serwera Keycloak który ma za zadanie zweryfikować autentyczność oraz wymienić klucz identyfikujący na klucz autoryzujący dający dostęp do serwera zasobów. Przebieg informacji został zobrazowany na ilustracji (Rysunek 3.2).

Autoryzacja użytkownika, czyli proces weryfikacji praw dostępu do zasobów odbywa się bez udziału zewnętrznych serwisów, natomiast bierze w nim udział inny aktor – serwer zasobów (REST API). Jest to czynność wykonywana przy każdej próbie pobrania danych i jest dwupoziomowa. Za pośrednictwem aplikacji mobilnej klient wykonuje żądanie sieciowe dołączając do niego w nagłówku klucz autoryzujący, uzyskany w procesie uwierzytelnienia. Pierwszy etap polega na jego weryfikacji. Po stronie serwera ta wartość jest wyodrębniana z zapytania sieciowego, a następnie przesyłana do oprogramowania Keycloak, gdzie odbywa się sprawdzenie jego poprawności. Pomyślne wykonanie tego etapu jest podstawą i wnosi niezbędne informacje



Rysunek 3.2: Przebieg informacji w procesie uwierzytelnienia użytkownika

do przeprowadzenia drugiego z nich. Zapisane w kluczu informacje pozwalają zidentyfikować użytkownika, który domaga się zasobów. Lecz istnieje niebezpieczeństwo, że dane o które klient wnioskuje nie należą do niego. Dlatego celem następnej weryfikacji jest umożliwienie dostępu tylko do tych zasobów, których użytkownik jest właścicielem. Często sprawdzane jest również to czy zasób jest też powiązany z firmą w kontekście której, wykonywane jest zapytanie. Wynika to z założenia, że dane są przechowywane indywidualnie dla każdej z firm. Oznacza to między innymi, że użytkownik prowadzący dwie firmy nie powinien mieć możliwości dodania do magazynu firmy „A” produktów stworzonych na podstawie modelu z firmy „B”. Są one całkowicie niezależne, a wykonywane operacje powinny odbywać się jedynie w kontekście jednej z nich. W takiej sytuacji weryfikacja odbywa się następująco:

Stan początkowy: Klient X wysłał żądanie w kontekście firmy Y o zasób Z.

1. Odszukaj zasób Z w bazie danych.
2. Przejdź po referencji w obiekcie zasobu do firmy, z którą jest powiązany.
3. Sprawdź czy firma związana z zasobem Z jest firmą Y. Jeżeli nie zakończ algorytm brakiem autoryzacji.
4. Przejdź po referencji w obiekcie firmy Y do użytkownika, który jest jej właścicielem.
5. Sprawdź czy właścicielem firmy Y jest użytkownik, który podpisał się kluczem autoryzującym. Jeżeli nie zakończ algorytm brakiem autoryzacji.
6. Udel autoryzacji.

Dzięki temu rozwiązaniu, system jest w stanie zweryfikować, czy klient powinien uzyskać dostęp do żądanego zasobów.



3.6 Uwierzytelnienie z wykorzystaniem zewnętrznych usług

Oprogramowanie do logowania i rejestracji wykorzystuje zewnętrzne usługi dostarczane przez firmę Google oraz Facebook. Ich wykorzystanie pozwala klientowi na dostęp do swojego konta wykonując zaledwie kilka kliknięć, a podczas kolejnych logowań istnieje możliwość automatycznego zalogowania bez żadnej interwencji użytkownika.

Nie bez znaczenia jest również fakt, że użytkownik systemu Android - na który tworzona jest aplikacja mobilna - już na etapie konfiguracji urządzenia jest proszony o utworzenie lub zalogowanie się na istniejące konto. Wykorzystuje się je m.in. w celu zapisania kopi systemu w chmurze oraz pozwala korzystać z usług i aplikacji Google. Urządzenia z systemem Android posiadają już fabrycznie zainstalowane programy takie jak mapy, poczta Gmail, czy też bardzo popularny YouTube. Każdy użytkownik systemu, a tym bardziej korzystający ze sklepu z aplikacjami Google, posiada konto. Dzięki temu nie pojawia się problem, w którym użytkownik próbujący uzyskać dostęp do oprogramowania, którego dotyczy niniejsza praca, jest zmuszany do założenia konta w jednym z zewnętrznych serwisów.

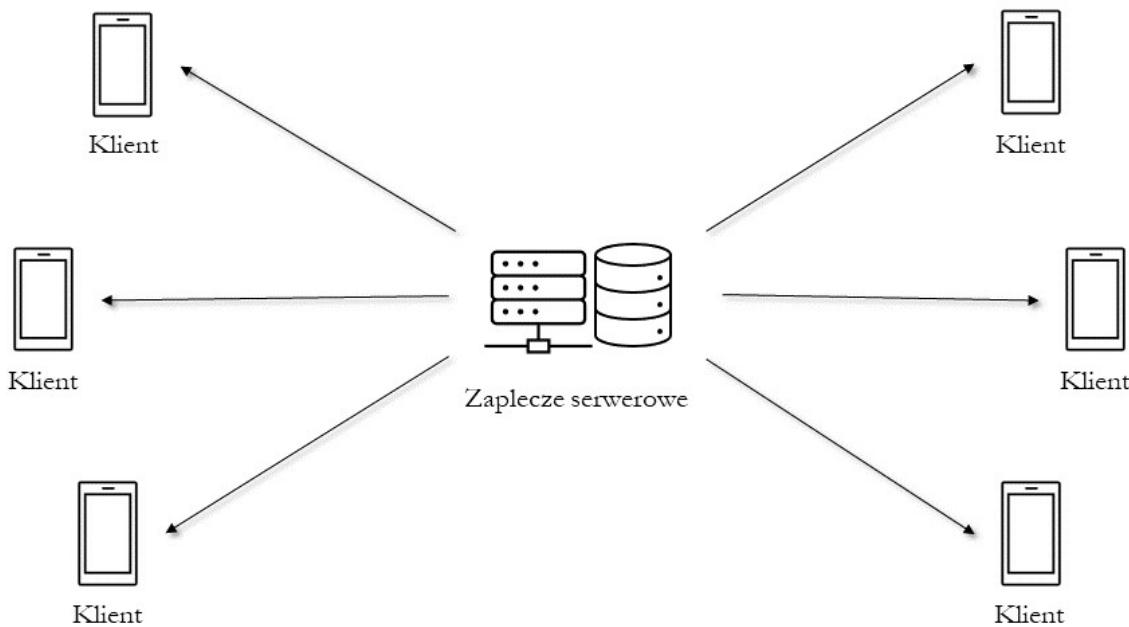
Zestaw, który tworzy metoda logowania z użyciem konta Google oraz Facebook jest popularnym rozwiązaniem stosowanym w wielu systemach. Rozpoznawalność marek jest gwarantem bezpieczeństwa pozyskiwanych danych. W istotny sposób wpływa na odbiór aplikacji przez użytkowników oraz sprawia, że aplikacja w odbiorze staje się bardziej atrakcyjna i profesjonalna.



Projekt systemu

W tym rozdziale przedstawiono projekt systemu zawierający opis modelu biznesowego. Za pomocą diagramu komponentów UML zaprezentowano strukturę systemu oraz występujące w nim zależności, a do opisu bazy danych wykorzystano jej graficzną reprezentację w postaci schematu.

4.1 Model biznesowy



Rysunek 4.1: Model biznesowy oprogramowania

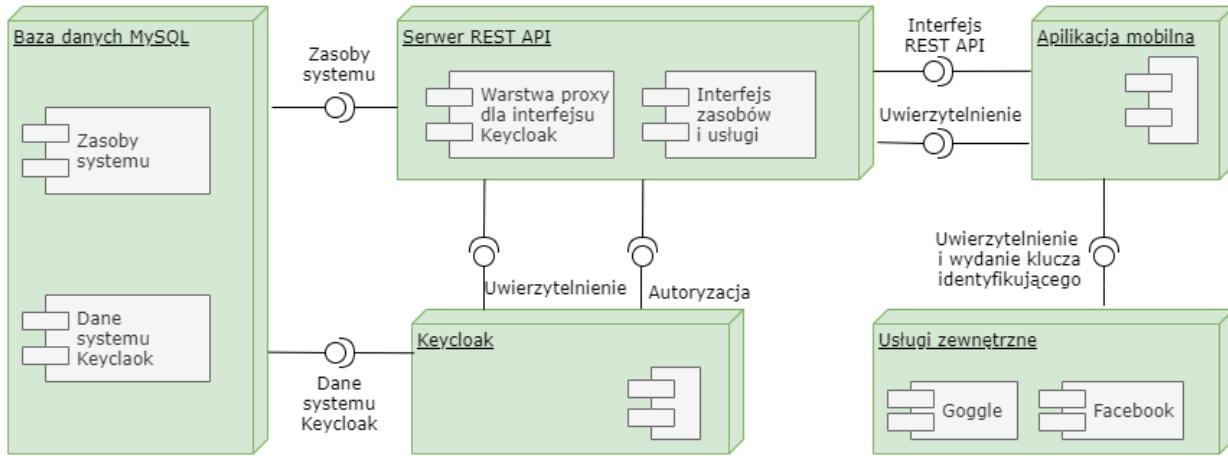
Analiza grupy docelowej dla której tworzone jest oprogramowanie, wymaga ustalenia najbardziej adekwatnego dla niej modelu biznesowego/sprzedażowego. Trzeba mieć na uwadze, że osoby prowadzące jedno lub kilkuosobową działalność często mają ograniczone zasoby zarówno sprzętowe, finansowe czy też merytoryczne. Nie każdy potencjalny klient posiada serwerownie, nie każdy chce ponosić co miesięczne koszty wynajmu wirtualnego serwera lub zakupu usługi utrzymania systemu. Nabywca nie musi być również specjalistą w dziedzinie informatyki, więc oprogramowanie nie powinno wymagać od niego jej znajomości lub zmuszać go do zatrudnienia osób trzecich.

Przyjęte rozwiązanie powinno być możliwie elastyczne i uniwersalne, co pozwoli stworzyć oprogramowanie nie wymagające dedykowanej konfiguracji. To z kolei sprawia, że system można oferować wielu klientom



i jednocześnie dystrybuować je w niższej cenie idealnie wpasowując się w wymagania rynku. Dlatego wybrano model (Rysunek 4.1), w którym system opiera się na jednej instancji oprogramowania serwerowego zarządzanego przez dystrybutora oprogramowania, a klient nabywa wyłącznie dostęp do aplikacji mobilnej. Potencjalnym kanałem udostępniania aplikacji jest sklep od firmy Google, który zapewnia możliwość przedstawienia oprogramowania dużej bazie odbiorców.

4.2 Diagram komponentów

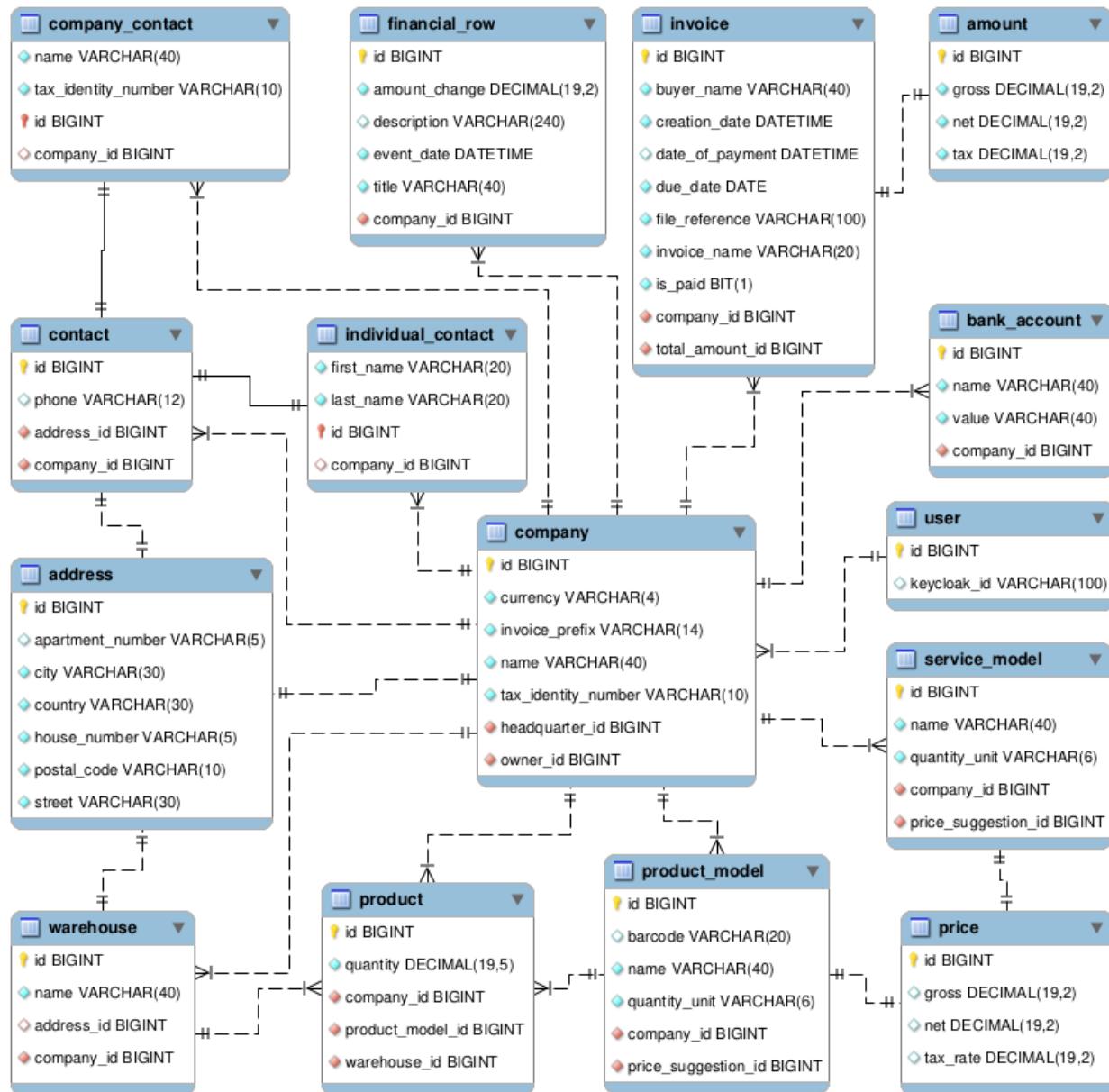


Rysunek 4.2: Diagram komponentów

System wspomagający zarządzanie biznesem opiera się o kilka najważniejszych komponentów przedstawionych na ilustracji (Rysunek 4.2). Interfejsem dostępnym dla użytkownika jest aplikacja mobilna. Za jej pośrednictwem klient jest w stanie zarządzać swoją firmą i kontem, przełączając się pomiędzy kolejnymi widokami. Aplikacja wykorzystuje usługi zewnętrzne od firm Google oraz Facebook, które odgrywają kluczową rolę w procesie uwierzytelniania. Są one dostarczycielem informacji o użytkowniku, dzięki czemu system jest w stanie go poprawnie rozpoznać i wydać mu odpowiedni klucz dostępu. Jej komunikacja z powiązanymi komponentami odbywa się za pośrednictwem sieci internetowej. Serwer REST API jest źródłem danych dla aplikacji mobilnej. Udostępnia interfejs dla klientów sieciowych pozwalający pobierać dane zapisane w systemie oraz odpowiada za przetwarzanie poleceń. Przykładem jest generowanie faktur w formacie PDF, któremu towarzyszą obliczenia finansowe. Jest odpowiedzialny za weryfikację uprawnień dających dostęp do danych, która składa się z dwóch etapów. Sprawdzana jest poprawność klucza którym podpisuje się użytkownik, oraz czy zasoby do których się odwołuje należą do niego. Jest też pośrednikiem w komunikacji z oprogramowaniem autoryzującym zapewniając prawidłowe działanie systemu i rozwiązując problem implementacyjny opisany w dalszej części pracy (Rozdział 5.3.1). Keycloak jest oprogramowaniem, które wydaje oraz sprawdza ważność kluczy. Jego głównym zadaniem jest bezpieczne powiązanie klienta z kontem w systemie. Ostatnim komponentem jest system do zarządzania bazą danych MySQL. Pełni on rolę magazynu informacji w którym przechowywane są dane pochodzące z dwóch powiązanych z nim komponentów.

4.3 Baza danych

Do realizacji niniejszej pracy wykorzystano otwarto źródłowy system do zarządzania relacyjnymi bazami danych MySQL[12]. Jest to jedno z najpopularniejszych rozwiązań dostępnych na rynku. Informacje zapisane w relacyjnej bazie danych są zorganizowane jako zbór relacji (tabel) opisanych pewnymi związkami, a w każdej



Rysunek 4.3: Schemat bazy danych

z nich znajdują się krotki (wiersze). Komunikacja klienta z bazą danych odbywa się z użyciem języka SQL (ang. Structured Query Language).

W trakcie działania systemu, często będą wykonywane zapytania sieciowe mające na celu pobranie danych z bazy odwołując się do identyfikatorów krotek. Zanim zasoby zostaną wydane klientowi, konieczna jest weryfikacja ich przynależności do wykonującego zapytanie użytkownika. Żądanie często jest wykonywane w kontekście konkretnej firmy, więc sprawdzane jest również to czy zasób do niej należy. Mechanizm polega na przejściu ścieżki od konkretnego zasobu - do którego użytkownik się odwołuje - poprzez firmę, do której jest przypisany, aż do użytkownika który jest właścicielem obu wcześniejszych danych. Na koniec system sprawdza czy właściciel danych jest użytkownikiem wykonującym żądanie oraz czy kontekst firmy jest prawidłowy. Z tego powodu krotki do których klient może się odwoływać po identyfikatorze za pomocą interfejsu REST API, powinny być powiązane z firmą oraz użytkownikiem jak najkrótszą ścieżką. To pozwoli na optymalizację



często wykonywanego procesu.

Aplikacja mobilna często żąda (pośrednio poprzez REST API) zbiorów wszystkich faktur, kontaktów, produktów, magazynów, itd. związanych z konkretną firmą. Z tego względu zastosowano optymalizację polegającą na powiązaniu tych danych bezpośrednim związkiem z relacją `company`.

Powyższe optymalizacje nie zakłócają się wzajemnie. Zawarcie klucza obcego w relacji `company` tworzącego związek z tabelą `user`, pozwala szybko zweryfikować właściciela danej firmy. Natomiast zamieszczenie klucza obcego w pozostałych tabelach - do których użytkownik może się odwoływać za pośrednictwem REST API - tworzy referencję do krotki w relacji `company` oraz jest rozwiązaniem drugiej i uzupełnieniem pierwszej optymalizacji.

Założenia bazy danych:

- Użytkownik może posiadać wiele firm którymi zarządza.
- Modele produktów i usług, produkty, magazyny, kontakty, zdarzenia finansowe, konta bankowe oraz faktury są przechowywane oddzielne dla każdej firmy i nie są bezpośrednio powiązane z użytkownikiem.
- Kontakty indywidualne oraz firmowe posiadają wspólną pulę identyfikatorów.
- Baza danych jest zoptymalizowana pod kątem procesu weryfikacji dostępu opisanego powyżej.
- Baza danych uwzględnia drugą opisaną powyżej optymalizację.

W bazie danych znajduje się 15 tabel widocznych na rysunku 4.3, są nimi:

- `user` – tabela wiążąca identyfikator użytkownika pochodzący z serwera autoryzującego z wewnętrznym identyfikatorem wykorzystywanym w systemie.
- `company` – tabela zawiera informacje o zarządzanych w aplikacji firmach. Te dane są rozszerzone o krotkę z tabeli `address` z którą zachodzi związek *jeden do jednego*. Każdy wiersz tabeli `company` jest powiązany zależnością *wiele do jednego* z rekordem w tabeli `user`. Dzięki temu użytkownik wykorzystując jedno konto może zarządzać wieloma firmami. Pozostałe związki typu *jeden do wielu* mają na celu powiązanie danych z konkretną firmą lub idąc dalej konkretnym użytkownikiem. Część z nich może wyglądać na zbędne, jednak są istotne w celu optymalizacji.
- `address` – tabela zawiera informację o adresach. Zachodzi tylko w związku typu *jeden do jednego*.
- `price` – tabela zawiera informację o cenie netto, brutto oraz stawce podatku.
- `service_model` – tabela przedstawia informację o usługach oferowanych przez będące z nimi w związkach firmy. Każda usługa jest powiązana z sugerowaną ceną zapisaną w tabeli `price`.
- `product_model` – tabela przedstawia informację o modelach produktów oferowanych przez będące z nimi w związkach firmy. Każdy z nich jest powiązany z sugerowaną ceną zapisaną w tabeli `price`.
- `warehouse` – tabela przedstawia informację o magazynach należących do powiązanych firm. Krotki są związane z tabelą `address` w której zapisana jest lokalizacja magazynu.
- `product` – tabela przedstawia informację o ilościach produktów w magazynach. Każdy rekord jest w związku z magazynem oraz modelem produktu.
- `bank_account` – tabela przedstawia informację o kontach bankowych (nazwę oraz numer) powiązanych z firmami.
- `amount` – tabela zawiera informacje o kwotach: netto, brutto oraz podatku
- `invoice` – tabela zawiera najważniejsze informacje o wystawionych w systemie fakturach. Ich opis zawiera m.in. numer, nazwę nabywcy, datę utworzenia czy referencję do dokumentu PDF. Wykorzystując klucz obcy, dostępne są również informacje o ich całkowitych wartościach.
- `financial_row` – tabela zawierająca informację o zdarzeniach finansowych powiązanych z firmą. Jej pola zawierają tytuł, kwotę zmiany, datę zdarzenia oraz opis.



- **contact** – tabela zawiera reprezentacje abstrakcyjnych kontaktów. W systemie jest to klasa nadzędna biorąca udział w mechanizmie polimorfizmu. Obrana strategia mapowania polega na łączeniu tabel. Klucze główne rekordu mają swój odpowiednik w jednej z tabel **individual_contact** lub **company_contact**
- **individual_contact** – tabela zawierająca imiona i nazwiska, czyli informacje charakterystyczne dla kontaktów prywatnych. W wyniku wewnętrznego połączenia - wykorzystując klucze główne - z tabelą **contact** powstaną kompletne rekordy reprezentujące kontakty prywatne.
- **company_contact** – tabela zawierająca nazwy oraz numery identyfikacji podatkowej. W wyniku wewnętrznego połączenia - wykorzystując klucze główne - z tabelą **contact** powstaną kompletne rekordy reprezentujące kontakty firmowe.



Implementacja systemu

W tym rozdziale zaprezentowano mechanizm autoryzacji użytkownika oraz zastosowany algorytm filtrowania treści. Zawarto w nim również opis środowiska programistycznego oraz wzorców projektowych i bibliotek, które wykorzystano do realizacji pracy. Rozwój oprogramowania ujawnił kilka problemów implementacyjnych. Podrozdział 5.3 przedstawia analizę wybranych z nich.

5.1 Środowisko programisty

Rozwój aplikacji mobilnej przebiegał z wykorzystaniem zintegrowanego środowiska programistycznego (ang. IDE) Android Studio. Jest to kompleksowe narzędzie z wbudowanym emulatorem urządzeń, na których istnieje możliwość testowania oprogramowania. Środowisko umożliwia również bezpośrednią instalację aplikacji na fizyczne urządzenia, bez konieczności tworzenia instalowalnych plików APK. Dobrze spełnia rolę debuggera, pozwalaając szybko rozpoznać pojawiające się błędy. Środowisko jest zintegrowane z systemami kontroli wersji oprogramowania, również z systemem Git, który jest wykorzystywany w niniejszej pracy [4].

Implementację serwera zasobów wykonano przy użyciu oprogramowania IntelliJ IDEA skonfigurowanego pracy z językiem Java w wersji 11. Jest to środowisko, na którym zostało zbudowane Android Studio. Z tego względu ich interfejs jest spójny podnosząc komfort rozwijania obu systemów jednocześnie (przyp. aplikacji mobilnej i serwera zasobów). Do rozwoju i przeprowadzania testów REST API wykorzystano aplikację Postman.

Bazą danych zarządzano wykorzystując oprogramowanie MySQL Workbench. Narzędzie rozwijane przez firmę Oracle Corporation służy do projektowania, tworzenia i administrowania bazami danych. Instancję stworzoną w ramach projektu wraz z serwerem zasobów oraz oprogramowaniem Keycloak umieszczono w kontenerze stworzonym z wykorzystaniem systemu Docker.

5.2 Mechanizm filtrowania treści

Części z list - znajdujących się w widokach interfejsu aplikacji mobilnej - towarzyszy panel wyszukiwania, pozwalający odfiltrować pozycje na podstawie wprowadzonego tekstu. Każdy z tych mechanizmów oparty jest o tę samą strategię. Filtrowane obiekty implementują specjalny interfejs, który pozwala pobrać z nich reprezentujący je ciąg tekstowy. Często jest to złożenie wszystkich dostępnych informacji dla danego obiektu, przykładem jest implementacja interfejsu dla kontaktów, która składa dane dotyczące nazwy (lub imienia i nazwiska w przypadku osoby prywatnej), numeru telefonu, adresu oraz numeru identyfikacji podatkowej, jeżeli taki posiada. Jednak istnieją też takie w których część pól jest pomijana, jak w przypadku produktów, ignorowana jest m.in. informacja dotycząca stawki podatku vat. Im lepsza reprezentacja obiektu tym łatwiejsze dla użytkownika staje się odszukanie interesującej go pozycji. Algorytm obranej strategii prezentuje się następująco.

1. Pobierz wprowadzony w pole wyszukiwania tekst.
2. Zamień wszystkie wielkie litery na małe.
3. Podziel tekst w miejscach wystąpienia spacji, tworząc listę.
4. Dla każdego filtrowanego obiektu:

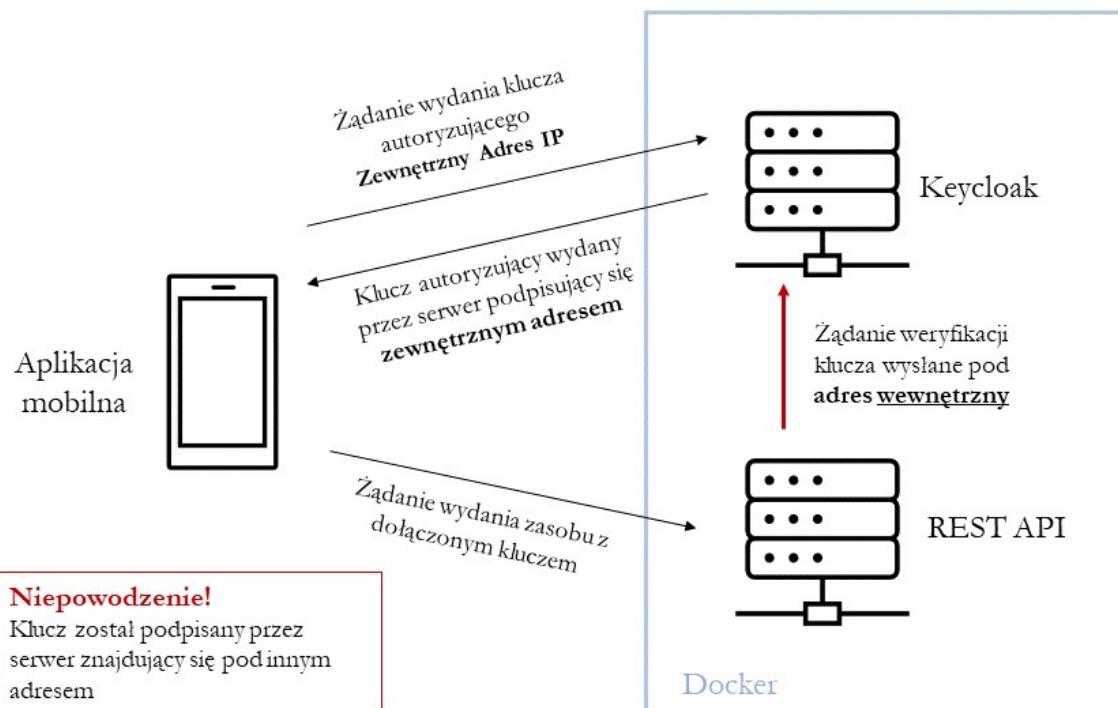
- Pobierz ciąg tekstowy reprezentujący obiekt.
- Zamień wszystkie duże litery na małe.
- Jeżeli wszystkie „słowa” z pkt. 3. są częścią przygotowanego ciągu tekstopowego, dodaj obiekt do listy, która zostanie wyświetlona po zakończeniu filtrowania.

Wykorzystanie takiego algorytmu pozwala przeszukać listę, ignorując wielkość liter i kolejność słów w reprezentacji obiektu. Umożliwia wyszukiwanie po wielu charakterystykach jednocześnie, tworząc rozwiązanie elastyczne.

5.3 Analiza problemów implementacyjnych

W tym podrozdziale opisano przyczyny i przedstawiono rozwiązania wybranych problemów implementacyjnych, które wstąpiły podczas realizacji systemu. W szczególności zwrócono uwagę na niepożądane wycieki pamięci oraz problem towarzyszący procesowi uwierzytelnienia.

5.3.1 Uwierzytelnienie użytkownika



Rysunek 5.1: Problem uwierzytelnienia użytkownika

Testy oprogramowania ujawniły nieoczekiwany problem. Zastosowanie konteneryzacji systemów składających się na zaplecze serwerowe sprawia, że istnieją dwie możliwości połączenia się z nimi za pomocą protokołów sieciowych. Kontenery w sieci wewnętrznej do komunikacji pomiędzy sobą korzystają z innych adresów niż w przypadku próby komunikacji z zewnątrz. W omówionym w rozdziale 3.5 procesie uwierzytelniania, drugi etap operacji polega na wymianie klucza identyfikującego na autoryzujący. Na rysunku 5.1 przedstawiony przy użyciu dwóch strzałek opisujących wymianę informacji pomiędzy aplikacją mobilną, a oprogramowaniem Keycloak. Wygenerowany klucz ma w swojej budowie zakodowaną informację o emitencie (ang. issuer). Jest



nią adres sieciowy którego domena i port są zgodne z tymi dostarczonymi przez klienta w żądaniu sieciowym. Klient łącząc się z serwerem autoryzującym z zewnątrz powoduje, że wydawcą wygenerowanego klucza jest adres zewnętrzny.

Weryfikacja klucza lub proces jego odświeżania, wykonywane bezpośrednio (z zewnątrz) na serwerze autoryzującym przebiega prawidłowo. Problem pojawia się, gdy następuje próba wykonania tych operacji z komponentu znajdującego się również wewnętrz kontenera Docker. Taka sytuacja zachodzi, gdy użytkownik zgłasza się po zasoby do serwera REST API. Przed ich wydaniem konieczna jest weryfikacja klucza, którym użytkownik podpisuje zapytanie sieciowe. Wykonywane w tym celu żądanie jest wysyłane pod adres wewnętrzny, różniący się od adresu zapisanego w kluczu. Mimo, że fizycznie serwerem wydającym i weryfikującym jest ta sama instancja oprogramowania, operacja zakończy się niepowodzeniem. System informuje, że nie może potwierdzić autentyczności klucza wydanego przez innego emitenta.

Rozwiązaniem omawianego problemu jest napisanie w sieci wewnętrznej, warstwy pośredniczącej (ang. proxy) w komunikacji sieciowej z oprogramowaniem Keycloak. Wykorzystanie jej przez klientów łączących się z kontenerem z sieci zewnętrznej pozwoli uniknąć problemu niepasujących do siebie adresów.

5.3.2 Procesy asynchroniczne oraz wycieki pamięci w aplikacji mobilnej

Największą część operacji asynchronicznych w aplikacji stanowią zapytania sieciowe. Są one wykonywane na oddzielnich wątkach, aby nie blokować głównego, który ma za zadanie dbać o responsywność graficznego interfejsu. Do ich obsługi wykorzystywany jest zestaw narzędzi. Wśród nich znajduje się biblioteka RxJava (ReactiveX dla języka Java), wprowadza ona programowanie reaktywne do języka, które jest rozszerzeniem wzorca projektowego obserwator. Połączenie programowania reaktywnego oraz charakterystyki działania aplikacji opartych na systemie Android generuje problemy.

Sposób w jaki korzysta się z aplikacji na urządzeniach mobilnych, charakteryzuje się tym, że wyświetlana treść jest bardzo często zmieniana. Części interfejsu użytkownika tworzą reprezentujące je *fragmenty*. Są to moduły, które zarządzają poszczególnymi widokami. To w nich są przechowywane referencje do komponentów, z których składa się poszczególny widok. Fragmenty mają swój cykl życia w systemie i w zależności od panującej sytuacji są m.in. tworzone, zatrzymywane, wznawiane lub niszczone. Przelaczanie *fragmentów* – np. wybranie innej pozycji w menu – powoduje, że następuje inicjalizacja nowego, a poprzedni zostaje zniszczony wraz z jego komponentami. W sytuacji, gdy istnieje aktywne powiązanie asynchronicznej aktywności z niszczonym komponentem, może się okazać, że w momencie jej zakończenia, *fragment* nie będzie już istniał, a odwołanie do niego zakończy się błędem. Najczęściej, wykonywane zapytania sieciowe są bezpośrednio lub pośrednio powiązane z komponentami wyświetlonymi w widoku na ekranie użytkownika. Poniżej znajduje się scenariusz, który na przykładzie przybliża opisywaną sytuację.

Stan początkowy: Użytkownik aplikacji znajduje się w *fragmencie*, który wyświetla listę kontaktów.

1. Użytkownik wybiera jeden z nich, aby zobaczyć jego detale. Jednocześnie tworzony jest nowy *fragment*, rozpoczęte jest asynchroniczne zapytanie sieciowe mające na celu pobranie potrzebnych informacji, a na ekranie wyświetlany jest symbol oznaczający ładowanie danych.
2. Użytkownik się rozmyśla i zanim rozpoczęte zapytanie sieciowe zakończy się odpowiedzią, wraca do poprzedniego widoku. Oznacza to, że *fragment* odpowiedzialny za prezentację szczegółów kontaktu zostaje zniszczony.
3. Nadchodzi odpowiedź na wykonane zapytanie sieciowe. Reagując na nią, aplikacja próbuje wyświetlić treści. Kończy się to błędem, ponieważ komponenty, do których te dane miały trafić, zostały zniszczone.

Innym pojawiającym się problemem są wycieki pamięci. Na przykładzie powyższego scenariusza można zauważyc, że po zmianie widoku zapytanie sieciowe, które nie będzie już wykorzystane jest ciągle aktywne. Wykonanie wielu takich operacji, którymi się nie zarządza i których się nie dezaktywuje w odpowiednim czasie, może prowadzić do ich namażania, co skutkuje niekontrolowanym użyciem pamięci.

Rozwiązaniem, dzięki któremu można zapobiegać powyższym problemom jest powiązanie aktywności asynchronicznych z cyklem życia *fragmentu*. Należy zadbać o to, aby w momencie, gdy użytkownik przełącza się pomiędzy widokami i pewne komponenty są niszczone, zaprzestać wykonywania operacji, które są z nimi powiązane. Takie możliwości daje wykorzystanie biblioteki RxLifecycle. Dzięki niej możliwe jest wkomponowanie aktywności w cykl życia *fragmentu*.



5.4 Biblioteki i narzędzia

Biblioteki są istotnym elementem projektów programistycznych. Ich wykorzystanie pozwala zrealizować system w krótszym czasie. Programista nie jest zmuszony pisać kodu, który został już napisany oraz minimalizuje możliwość popełnienia błędów. Biblioteki często realizują złożone problemy, a ich implementacja jest czasochłonna. Wykorzystanie istniejących rozwiązań pozwala skupić się na indywidualnych elementach oprogramowania. W tym podrozdziale przedstawiono narzędzia, które zastały wykorzystane do implementacji systemu, którego dotyczy niniejsza praca.

5.4.1 Dagger

Do wstrzykiwania zależności w aplikacji mobilnej wykorzystano bibliotekę Dagger^[2]. Jest to platforma programistyczna dla języka Java i Kotlin, która procesuje adnotacje i generuje kod źródłowy Java. Przenosi ona odpowiedzialność za tworzenie obiektów na kontener. W celu powiązania obiektów posługuje się konfiguracją, która określa jak obiekty powinny być powiązane. Graf zależności jest wyliczany w momencie komplikacji i na jego podstawie następuje inicjalizacja obiektów w odpowiedniej kolejności. Pozwala zredukować występujące w systemie zależności (ang. low Coupling), umożliwiając łatwiejsze testowanie i modyfikację fragmentów kodu.

5.4.2 Google Mobile Vision

Google Mobile Vision to zaawansowana biblioteka do analizy obrazów. Udostępnia aplikacyjne interfejsy do odnajdywania twarzy i analizy jej mimiki, potrafi rozpoznać uśmiech, mrugnięcia oczu i wiele więcej. Inna część biblioteki jest odpowiedzialna za rozpoznawanie tekstów na grafikach oraz udostępnia szereg narzędzi z nimi związanych. Projekt pozwala również, rozpoznawać kody kreskowe zarówno jedno- i dwuwymiarowe w różnych formatach. Analizuje wartości oraz automatycznie rozpoznaje i rozróżnia typy takie jak adres url, wizytówka kontaktu, email, telefon, ISBN, dane wifi czy pozycje geolokacyjną. Dokumentacja biblioteki jest zrealizowana w przyjazny sposób i zawiera instrukcję oraz przykłady pomagające w wdrożeniach ^[5].

5.4.3 Gson

Biblioteka dla języka Java - Gson - jest rozwijana przez firmę Google i jest przeznaczona do serializacji obiektów do formatu JSON i deserializacji danych w formacie JSON do obiektów Java. Projekt jest otwarto źródłowy, a sam producent udostępnia kod źródłowy biblioteki^[6]. Obsługuje kolekcje, klasy generyczne i polimorficzne oraz udostępnia wiele opcji konfiguracji narzędzia m. in. sposób obsługi pól z wartością `null`, czy mapowanie ich nazw.

5.4.4 Lombok

Lombok to projekt wspierający programistę w rozwijaniu oprogramowania z wykorzystaniem języka Java. Jest procesorem adnotacji, który przed komplikacją generuje dodatkowy kod. Pozwala on ograniczyć konieczność pisania powtarzalnych fragmentów kodu i zaoszczędzić tysiące lini kodu. Generuje dla klas często pisane konstruktory, gettery i settery. Lombok implementuje również metodę `equals()` umożliwiającą porównanie obiektów ^[10]. W trakcie testów oraz rozwoju oprogramowania użyteczna będzie metoda `toString()` reprezentująca obiekt w czytelny sposób.

5.4.5 Material Design

Opracowany przez główny zespół inżynierów i projektantów firmy Google styl graficzny – Material Design - jest rozwijany od 2014 roku. Towarzyszy nie tylko wszystkim internetowym i mobilnym produktom jej producenta, ale również wielu systemom opracowanym przez firmy niezależne. Biblioteka udostępnia duży zbiór komponentów graficznych, a na stronie projektu można znaleźć ich czytelną dokumentację^[11].



5.4.6 iText

Portable Document Format (PDF) został stworzony przez firmę Adobe Systems, a obecnie jest rozwijany przez Międzynarodową Organizację Normalizacyjną. Celem stworzenia formatu była eliminacja problemów związanych z wyświetlaniem treści dokumentów w różny sposób przez różne czytniki. Przedstawienie standardu spowodowało, że dokumenty niezależnie od urządzenia, wykorzystanego oprogramowania czy systemu operacyjnego są wyświetlane w ten sam sposób.

Istnieje kilka metod tworzenia plików PDF. W procesie ręcznym można wykorzystać aplikację Adobe Acrobat, jednak takie rozwiążane nie jest stworzone do generowania dużej ilości plików automatycznie. Innym sposobem jest wykorzystanie szablonów np. XML, HTML, które następnie są konwertowane do docelowego formatu. Jego wadą jest mała elastyczność. Na przykładzie generowania faktur, problemy stwarzają niewielka liczba pozycji na etapie tworzenia szablonu. Istnieje możliwość dynamicznego kopiowania wierszy, aby ich liczba była zgodna, jednak nie jest to jedyna komplikacja, a sam sposób rozwiązania problemu budzi wątpliwości co do stylu (dokument nie jest tworzony bezpośrednio w docelowym formacie). Zastosowaną w projekcie metodą jest wykorzystanie biblioteki iText, która pozwala bezpośrednio z kodu źródłowego dowolnie kreować dokumenty [7]. Jest dostępna dla wielu języków programowania w tym języka Java. Dzięki jej wykorzystaniu możliwe jest dynamiczne i automatyczne tworzenie dokumentów PDF. Wśród zalet jest również niska waga wynikowych plików.

5.4.7 Spring

Spring to narzędzie złożone z wielu projektów, które są przeznaczone do tworzenia oprogramowania w języku Java i jest dobrze opisane na stronie producenta[16]. Kluczowym jego elementem jest udostępniana platforma programistyczna (ang. framework), która opiera się na dwóch podstawowych funkcjach. Pierwszą z nich jest kontener IoC (Inversion of Control). Jest on odpowiedzialny za zarządzanie ziarnami (ang. Bean), czyli obiektami kontrolowanymi przez kontekst Spring. Przeniesienie odpowiedzialności tworzenia obiektów na kontener, nazywane jest wzorcem odwróconego sterowania. Druga funkcja – wstrzykiwanie zależności (ang. Dependency Injection) - jest jednym ze sposobów jego realizacji.

Inną wykorzystywaną częścią platformy jest Spring Web MVC. Udostępnia ona adnotacje, umożliwiające tworzenie kontrolerów zapytań sieciowych. Do jej kluczowych funkcjonalności należy między innymi mapowanie żądań, które umożliwia odwzorowanie zapytań sieciowych na konkretne metody kontrolerów.

Projektem wspierającym integracje z bazą danych jest Spring Data JPA. Głównym jego celem jest zredukowanie ilości powtarzającego się kodu. Wykorzystanie go w systemie, ułatwia uzyskać dostęp oraz w większości przypadków eliminuje potrzebę pisania zapytań do bazy danych. Tworzenie repozytoriów najczęściej sprowadza się do napisania interfejsu z metodami nazwanymi według ustalonej konwencji. Dzięki temu Spring jest w stanie automatycznie wygenerować jego implementację.

Projekt Spring Security pozwala zabezpieczyć aplikację internetową oraz stworzyć system autoryzacji i uwierzytelnienia użytkowników. Dostarcza gotowe mechanizmy w wyniku czego programista jest zwolniony z tworzenia rozwiązań od podstaw. Umożliwia zabezpieczenie interfejsu REST API oraz integrację z oprogramowaniem Keycloak.

Zrealizowany system wykorzystuje Spring Boot, który pozwala na szybką konfigurację projektów (wchodzących w skład Spring), bez którego może się ona okazać czasochłonna i skomplikowana. Rozwiązanie jest oparte o przygotowaną kolekcję szablonów startowych. Dzięki czemu programista nie jest zobowiązany do manualnego dołączania i integracji wymaganych zależności.

5.4.8 Retrofit

Retrofit jest rozbudowaną biblioteką, która umożliwia komunikację z serwerem REST. Zamienia interfejs HTTP na obiekty języka Java. Wykorzystanie jego interfejsu pozwala na dużą konfigurację: zastosowanie konwerterów czy też różnych klientów sieciowych. Poprzez dodanie odpowiedniego adaptera współpracuje z biblioteką ReactiveX, a odbierane odpowiedzi na wykonane zapytania sieciowe są automatycznie mapowane z formatu JSON na obiekty. Retrofit udostępnia funkcjonalny interfejs, który wraz z przykładami jest opisany na stronie projektu[14].



5.4.9 ReactiveX

ReactiveX jest biblioteką, która łączy najlepsze rozwiązania wzorca projektowego obserwator oraz programowania funkcyjnego [13]. Umożliwia asynchroniczne oraz bazujące na zdarzeniach (ang. events) programowanie. Działanie polega na przetwarzaniu danych, które są reprezentowane w postaci strumienia **Observable** i obsługiwane przez obiekty **Observer**. Pojawiające się w nich asynchroniczne dane mogą być agregowane, filtrowane i mapowane, a całe strumienie ze sobą kombinowane.

5.4.10 RxLifecycle

Zapobieganie wyciekom pamięci, które są spowodowane aktywnymi subskrypcjami na niezakończonych strumieniach danych **Observable**, zostało zrealizowane z wykorzystaniem projektu RxLifecycle. Dostarczone przez bibliotekę mechanizmy, pozwalają powiązać wykonywanie operacji z cyklem życia *fragmentów* [15]. Wkomponowanie ich w strumień danych najczęściej odpowiada za jego zamknięcie, gdy widok będzie niszczony. Jednak narzędzie daje również możliwość powiązania go z innymi stanami cyklu życia.

5.5 Wykorzystane wzorce projektowe

Stosowane w systemach wykorzystujących programowanie obiektowe - wzorce projektowe, są zestawem sprawdzonych rozwiązań do często występujących problemów programistycznych. W tym podrozdziale przedstawiono te z nich, które znalazły zastosowanie w implementowanym systemie.

5.5.1 Wstrzykiwanie zależności

Wzorzec projektowy wykorzystywany zarówno w serwerze REST przy pomocy Spring oraz w aplikacji mobilnej z użyciem biblioteki Dagger. Wstrzykiwanie obiektów pozwala wyeliminować bezpośrednie zależności pomiędzy elementami systemu. Jest on jednym ze sposobów realizacji paradygmatu odwróconego sterowania (ang. Inversion of Control), który polega na przeniesieniu odpowiedzialności za sterowanie wykonywaniem programu do platformy programistycznej (ang. framework). Spring stosuje kontener odpowiedzialny za tworzenie i zarządzanie obiekttami, który jest wykorzystywany do wstrzykiwania zależności.

5.5.2 ViewHolder

Wzorzec kreacyjny ViewHolder ma na celu optymalizację procesu wyświetlania komponentów na widoku. Jest szczególnie istotny w przypadku wyświetlania kolekcji elementów na listach. Usprawnienie odbywa się poprzez ograniczenie liczby kosztownych operacji wiązania widoków z reprezentującymi je obiekttami. Do realizacji celu wykorzystuje się obiekt **ViewHolder**, który przetrzymuje referencje do instancji widoków. Dzięki temu, proces wiązania widoku z obiektem odbywa się tylko za pierwszym razem. W kolejnych sytuacjach wykorzystuje się, zapisaną referencję.

5.5.3 Obserwator

Obserwator to wzorzec projektowy umożliwiający obiektom powiadamianie innych obiektów o zmianie swojego stanu. Często jest wykorzystywany w sytuacjach, gdy pewien proces oczekuje na dane z innych asynchronicznych operacji, aby w pewien sposób je obsłużyć, ale nie wie w którym momencie informacje będą dostępne.

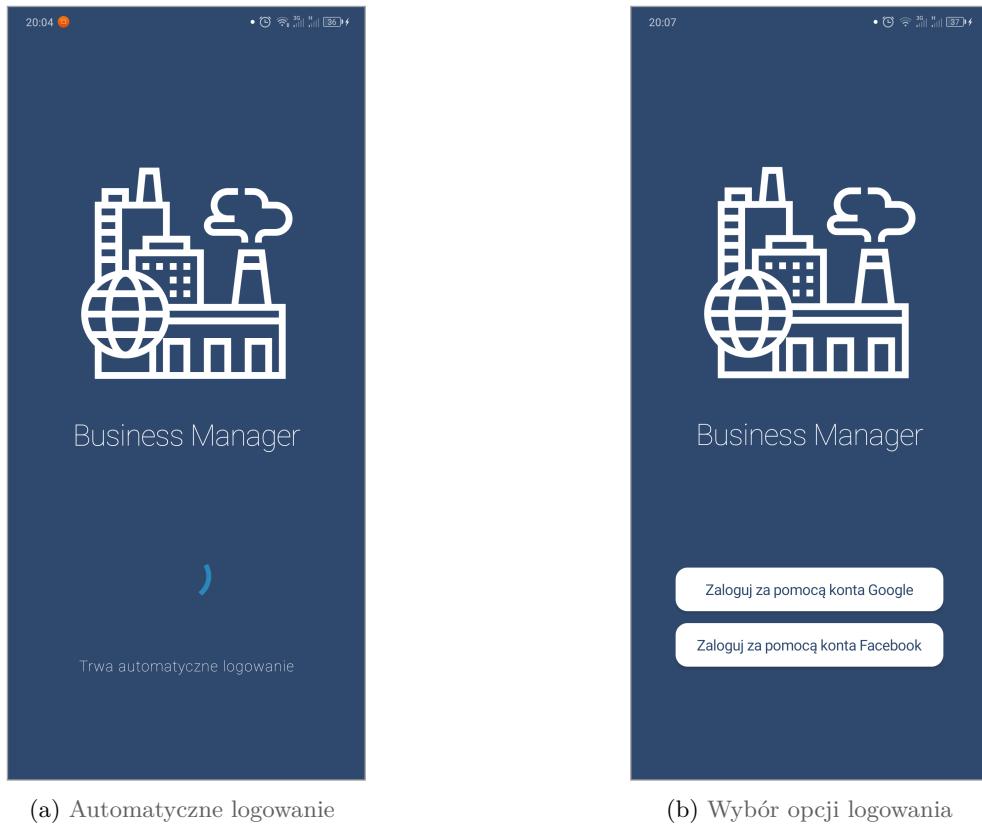
5.5.4 Model Widok Prezenter

Jednym z wzorców architektonicznych jest Model Widok Prezenter. Służy on do organizowania struktury programów posiadających graficzny interfejs użytkownika. Składa się z trzech głównych elementów. „Model” jest reprezentacją modułu, z którego czerpane są dane. „Widok” odzwierciedla widok oraz interfejs użytkownika. Natomiast „Prezenter” jest reprezentacją modułu zawierającego logikę aplikacji. Obsługuje żądania użytkownika oraz pobiera dane z „Modelu”, przetwarza i przekazuje do „Widoku”.

Interfejs użytkownika

Warunkiem powodzenia systemu na rynku, niejednokrotnie okazuje się dobrze zaprojektowany interfejs użytkownika. Jego funkcjonalne wykonanie pozwala na szybki dostęp do każdej części systemu. Wykorzystanie ikon poprawia czytelność aplikacji, a mechanizmy wyszukiwania i sugerowania podpowiedzi, pozwalają zaoszczędzić czas. W wielu sytuacjach zapobiega wykonaniu niedozwolonych operacji i wprowadzeniu nieprawidłowych danych. W tym rozdziale przedstawiono graficzny interfejs użytkownika. Zawiera opisy i rysunki dotyczące widoków znajdujących się w aplikacji mobilnej.

6.1 Widok ekranu logowania



Rysunek 6.1: Widoki ekranu logowania

Uruchomienie aplikacji powoduje wyświetlenie użytkownikowi ekranu powitalnego (ang. Splash screen). W jego centralnej części znajduje się grafika wraz z nazwą aplikacji, którą zobaczyć możemy na obu powyższych zrzutach. Po upływie 1,5 sekundy rozpoczyna się animacja, której celem jest płynne przejście do ekranu logowania. W jej wyniku dotychczas widoczny komponent jest przesuwany w górę, a poniżej



stopniowo – poprzez regulację przezroczystości – wyłania się jeden z dwóch komponentów widocznych na rysunkach 6.1a i 6.1b.

Pierwszy z nich jest wyświetlany w momencie, gdy użytkownik aktywuje w ustawieniach opcję automatycznego logowania. Aplikacja podejmuje próbę uwierzytelnienia, jednocześnie prezentuje na ekranie informację o zachodzącym procesie.

Jeżeli próba zakończy się niepowodzeniem lub wymieniona opcja nie zostanie aktywowana, użytkownik ma do swojej dyspozycji dwa przyciski które, są widoczne na drugim rysunku. Wybór jednego z nich uruchamia komponent - dostarczony przez wybranego usługodawcę - pozwalający na wybór konta lub ewentualne zalogowanie się. W przypadku użycia konta po raz pierwszy, użytkownik jest proszony o udzielenie zgody na przekazanie informacji potrzebnych do procesu uwierzytelnienia. Wciśnięcie jednego z przycisków powoduje, że zostaną zastąpione stosowną informacją wraz z animacją "paska postępu".

6.2 Widoki główne

(a) Lista zarządzanych firm

Firma	Stan finansów	Wartość produktów
Flamingo Inc. 37-021 Radom, Poland	229,032.47 PLN Stan finansów	34,649.32 PLN Wartość produktów
Netasca z.o.o 50-001 Wrocław, Polska	4,661.74 PLN Stan finansów	0.00 PLN Wartość produktów

DODAJ FIRMĘ

(b) Pulpit główny

Kategoria	Wartość	Detal
Faktury	3/6 Nieopłacone / Oplacone	9,338.50 PLN Nieopłacona wartość
Finanse	259.99 PLN Rozchody	229,292.46 PLN Przychody
Zasoby	4,999.99 PLN Ostatnie zdarzenie	229,032.47 PLN Bieżący stan
	2 Magazyny	34,649.32 PLN Wartość produktów

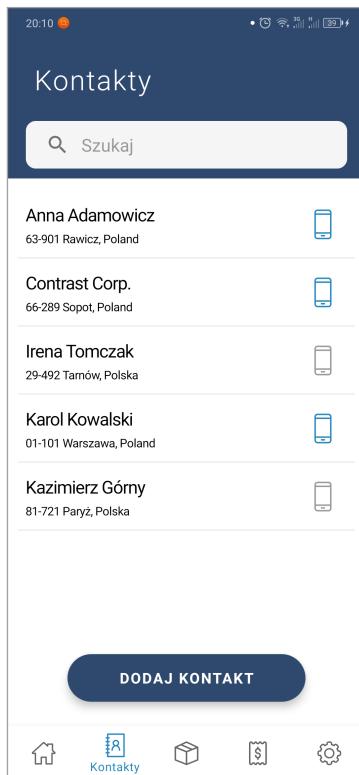
Pulpit

Rysunek 6.2: Widoki główne

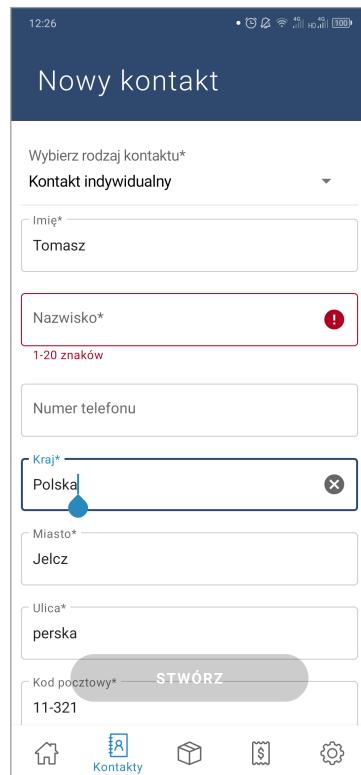
Po pomyślnie zakończonym procesie logowania, aplikacja kieruje użytkownika do ekranu, w którym prezentowana jest lista firm zapisanych na koncie (Rysunek 6.2a). Każdy z nich towarzyszą podstawowe informacje. Są nimi: adres siedziby firmy, stany finansów według prowadzonego w aplikacji rejestru oraz wartość produktów w magazynach. Na dole widoku znajduje się tak zwany płynący przycisk (floating button), czyli element, który znajduje się ponad pozostałą treścią. Na prezentowanym ekranie oznacza to, że wycinek listy - rozciągającej się do samego dołu ekranu - który w trakcie jej przewijania pokryje się z przyciskiem, zostanie przez niego zasłonięty. Jednak w trakcie implementacji zadbane o dodanie dolnego odstępu (ang. padding), dzięki któremu przewinięcie listy na sam dół nie spowoduje, że pewna treść nie będzie widoczna. Kliknięcie w przycisk spowoduje przekierowanie użytkownika do formularza, dzięki któremu będzie w stanie dodać do swojego konta nową firmę.

Wybór jednej z pozycji, spowoduje przejście do nowej aktywności. Jej graficznym elementem, który jest wyświetlany niezależnie od aktywnego fragmentu znajduje się na dole ekranu. Tak umieszczone i zbudowane z pięciu elementów menu, jest ergonomicznym rozwiązaniem pozwalającym na nawigację do głównych części systemu. Na rysunku 6.2b przedstawiono pierwszą z nich. Jest nią widok podsumowania, który składa się z trzech kart zatytułowanych odpowiednio faktury, finanse, zasoby. Każda z nich posiada cztery najistotniejsze statystyki związane z daną grupą. Wyświetlane informacje są przygotowywane i pobierane z dedykowanego interfejsu znajdującego się na serwerze. Zastosowanie takiego rozwiązania w stosunku do wykonywania obliczeń po stronie klienta, pozwala zredukować liczbę wykonywanych zapytań sieciowych.

6.3 Widok kontaktów



(a) Lista kontaktów firm



(b) Widok formularza dotyczący kontaktu

Rysunek 6.3: Widoki kontaktów

Kolejną pozycją dostępną w menu są kontakty. Zaletą ich przechowywania jest dostępność kontaktów na każdym urządzeniu, na którym użytkownik ma zainstalowaną aplikację. Istnieje również możliwość wykorzystania ich w procesie wystawiania faktur. Pojawiają się jako sugestie oraz pozwalają automatycznie uzupełnić formularz nabywcy i odbiorcy. W aplikacji przechowywane są dwa typy kontaktów, różniące się opisującymi je polami. Są to osoby prywatne oraz firmy. Reprezentujące je klasy wykorzystują polimorfizm rozszerzając wspólny interfejs. Dzięki temu, przedstawiony na rysunku 6.3a widok prezentuje listę wszystkich kontaktów niezależnie od ich typu.

Każdy wiersz zawiera nazwę oraz adres w skróconej formie. Po prawej stronie znajduje się ikona smartfona, która poprzez zastosowanie różnych kolorów informuje użytkownika o dostępności funkcji. Ukryta pod nią akcja odpowiada za wykorzystanie przypisanego do kontaktu numeru telefonu. Przenosi użytkownika do dialera telefonicznego z już wybranym numerem. Z tego miejsca do wykonania połączenia, dzieli go już tylko wcisnięcie słuchawki. Kliknięcie w pozostałą część wiersza spowoduje przejście do widoku detali odpowied-



niego dla typu kontaktu. Na tym ekranie użytkownik może zobaczyć jego szczegóły, usunąć go lub rozpocząć modyfikację.

W górnej części widoku znajduje się pole tekstowe udekorowane charakterystyczną ikoną lupy. Jest to panel wyszukiwania pozwalający odfiltrować listę kontaktów na podstawie wprowadzonego tekstu.

Znajdujący się na dole przycisk nawiguje do widoku zaprezentowanego na rysunku 6.3b. Jest to formularz, który pozwala dodać nowy kontakt. W pierwszym wierszu użytkownik korzystając z selektora wybiera typ kontaktu. Od wartości tego pola zależy widok pozostałych elementów formularza. Charakterystycznymi wierszami dla kontaktu indywidualnego są imię oraz nazwisko, a dla firmy nazwa oraz numer identyfikacji podatkowej.

Poprawność pól jest monitorowana oraz od niej zależy możliwość kliknięcia w przycisk znajdujący się na dole ekranu. Od czasu uruchomienia formularza do pierwszej edycji wartości konkretnego pola, nieprawidłowości nie są sygnalizowane. Powodami są wzgłydy estetyczne oraz sprzyjanie czytelności interfejsu. Pominięcie tej kwestii skutkowałoby tym, że od początku większość z pól byłaby oznaczona kolorem czerwonym, w sposób widoczny na rysunku dla pola **Nazwisko**. W tym wypadku zostało one podświetlone ze względu na to, że użytkownik rozpoczął wypełnianie tego pola jednak usunął całą treść i przeszedł do dalszej części formularza. Poza specjalnym kolorem pojawiła się krótka informacja, mówiąca o tym, w jaki sposób pole powinno zostać wypełnione. Aktualnie wybrany wiersz jest oznaczony kolorem niebieskim, a na jego prawej stronie dostępny jest przycisk pozwalający jednym kliknięciem wyczyścić wartość pola.

Poprawne wypełnienie formularza i zaakceptowanie jego treści poprzez kliknięcie w przycisk spowoduje zapisanie kontaktu na serwerze oraz przekierowanie użytkownika do widoku jego szczegółów.

6.4 Widok zasobów i produktów w magazynie

Zasoby

6 Modele produktów	4 Uslugi
-----------------------	-------------

Drukarnia
44-203 Radom, Poland (i)

Piwnica
23-429 Radom, Poland (i)

DODAJ MAGAZYN

Home Warehouse Products Currency Settings

Produkty

Szukaj	
Nazwa produktu	599.99 PLN
M.ZUIKO ED 60mm f/2.8 (czarny)	2 szt.
<hr/>	
Nazwa produktu	3,899.99 PLN
Olympus M-5 Mark II Body	4 szt.
<hr/>	
Nazwa produktu	254.99 PLN
Yongnuo YN 50mm f/1.8 Nikon	3 szt.
<hr/>	
Nazwa produktu	254.99 PLN
Yongnuo YN 50mm f/1.8 Nikon	67 szt.

DODAJ PRODUKT

Home Warehouse Products Currency Settings

(a) Widok głównego ekranu zasobów

(b) Widok produktów w magazynie

Rysunek 6.4: Widoki kontaktów

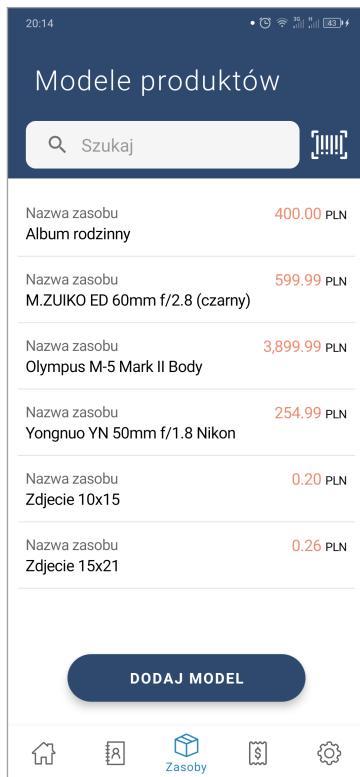
Centralna pozycja w menu prowadzi użytkownika do ekranu widocznego na rysunku 6.4a. Jest to punkt wyjściowy dla różnych części interfejsu odpowiedzialnych za zasoby firmy. Na górze znajdują się dwie umieszczone obok siebie karty. Wyświetlają one odpowiednio liczbę zadeklarowanych modeli produktów oraz liczbę dodanych przez użytkownika usług świadczonych przez firmę. Każda z nich jest również przyciskiem nawigującym do dwóch osobnych miejsc w systemie (rozdział 6.5) umożliwiając użytkownikowi zarządzanie tymi modelami. Poniżej znajduje się lista magazynów. Obok podstawowych informacji w wierszach, znajdują się specjalne ikony, które prowadzą do interfejsu prezentującego dane na temat konkretnego magazynu. Kliknięcie na pozostałą część wiersza spowoduje przejście do widoku, który reprezentuje rysunek 6.4b.

Jest to miejsce, w którym klient może prowadzić spis produktów znajdujących się w danym magazynie. Znajdujące się tu pozycje, są relacją modelu produktu oraz ich ilości. To właśnie na ich podstawie, z wykorzystaniem sugerowanej ceny jest obliczana wartość produktów w magazynach. Wybranie jednej z pozycji umożliwia zmianę jej stanu magazynowego, a także usunięcie konkretnego wiersza.

W części tytułowej, obok pola tekstowego pozwalającego na przeszukiwanie listy, znajduje się charakterystyczny przycisk, który pozwala na uruchomienie skanera kodów kreskowych. Jest on otwierany w oknie dialogowym, gdzie wyświetlany jest podgląd obrazu z kamery. Uzyskany w ten sposób kod zostaje wykorzystany do odnalezienia produktu, który jest z nim powiązany.

Znajdujący się na dole ekranu przycisk kieruje użytkownika do miejsca, w którym może dodać nowy produkt, wybierając jeden z zadeklarowanych modeli oraz uzupełniając go o ilość.

6.5 Widok modeli produktów i usług



(a) Widok listy modeli produktów



(b) Widok detali modelu produktu

Rysunek 6.5: Widoki modeli produktów

Wybranie jednej z dwóch krat opisanych w poprzednim rozdziale kieruje użytkownika do bliźniaczych widoków. Jednym z nich jest ekran modeli produktów widoczny na rysunku 6.5a. W porównaniu do widoku

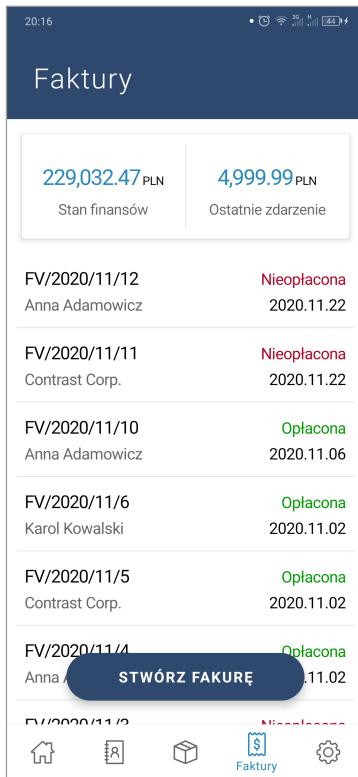


z listą usług, rozróżnia go przede wszystkim przycisk otwierający skaner kodów kreskowych, który tam nie występuje, a wraz z komponentem do tekstopowego filtrowania tworzy kompleksowe i intuicyjne rozwiązanie, usprawniające wyszukiwanie.

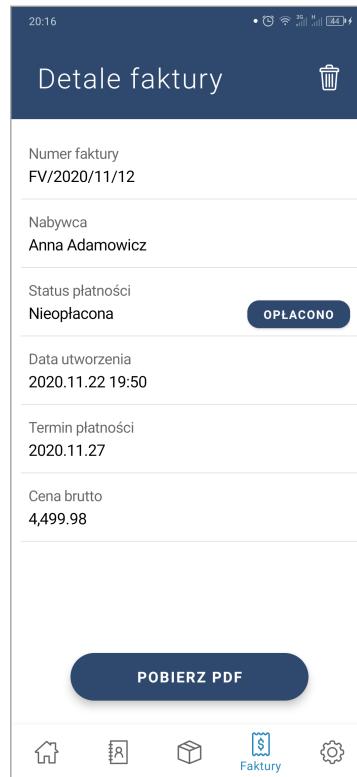
Modele zarówno usług jak i produktów są wykorzystywane w aplikacji do wspierania użytkownika w procesie tworzenia faktur. Poza tym drugie z nich są podstawą do uzupełniania spisu produktów w magazynach. Dodawanie nowego modelu odbywa się za pośrednictwem formularza dostępnego po naciśnięciu przycisku. Cechuje go komponent odpowiadający za wprowadzanie sugerowanej ceny produktu. Jest to zestaw trzech tekstowych elementów: ceny netto, stawki vat oraz ceny brutto, w którym ostatni z nich ma charakter *tylko do odczytu*. Jest w nim wyświetlana wartość obliczana i aktualizowana natychmiast po każdej zmianie dowolnej z pozostałych wartości.

Wybór jednej z dostępnych na liście pozycji, kieruje użytkownika do widoku detali przedstawionego na rysunku 6.5b. W części tytułowej znajduje się kosz pozwalający użytkownikowi usunąć obiekt. Główną treścią ekranu jest zestaw charakterystyk modelu. Cechą szczególną wyróżniającą ten typ widoków w aplikacji jest to, że gdy jedno lub kilka z pól jest niewypełnione, cały wiersz staje się niewidoczny. Powodem takiego zachowania jest zapobieganie złemu wrażeniu, które towarzyszy wyświetleniu pustych pozycji.

6.6 Widok listy faktur i ich szczegółów



(a) Widok listy wystawionych faktur



(b) Widok szczegółów faktury

Rysunek 6.6: Widoki dotyczące wystawionych faktur

Prowadzenie rejestru finansów oraz wystawianie faktur są jednymi z najistotniejszych funkcjonalności dostępnych w aplikacji. Dostęp do sekcji, która za nie odpowiada, odbywa się poprzez wybranie czwartej z pozycji dostępnych w menu. Głównym widokiem dla tej części interfejsu jest widok przedstawiony na rysunku 6.6a. Ważnym elementem jest znajdująca się na górze strony karta. Przedstawione na niej informacje dotyczą finansów firmy. Pierwszą z nich jest aktualny bilans środków, natomiast drugą informacją jest wartość ostatniego zarejestrowanego zdarzenia. Karta jest również przyciskiem, który nawiguje użytkownika do

rejestru finansów. Na dole ekranu znajduje się przycisk, dzięki któremu użytkownik może rozpoczęć proces generowania nowej faktury, który szerzej opisano w rozdziale 6.8. W jego tle widnieje lista utworzonych wcześniej dokumentów. Każdy wiersz jest reprezentacją pojedynczego obiektu. Widoczne informacje odnoszą się do numeru faktury, nazwy nabywcy znajdującej się na dokumencie oraz daty jego utworzenia. System wspomaga również monitorowanie jej stanu, z tego względu możliwe jest też zaprezentowanie adekwatnej informacji wyróżnionej odpowiednim kolorem w prawy górnym rogu wiersza.

Wybierając jedną z dostępnych pozycji, użytkownik zostaje przekierowany do widoku detali faktury zaprezentowanego na rysunku 6.6b. Na ekranie dostępne są trzy przyciski. Pierwszy – kosz – znajduje się w części tytułowej i umożliwia usunięcie obiektu z bazy danych oraz powiązanego dokumentu z pamięci serwera. Kolejny znajduje się obok statusu płatności. Przycisk jest widoczny tylko dla nieopłaconych faktur. Pozwala on zmienić jej status na opłaconą. Akcji towarzyszy dodatkowa funkcjonalność, która w momencie tej zmiany tworzy nowy przychód w rejestrze zdarzeń finansowych o wartości, na którą faktura została wystawiona. Ostatni znajdujący się na dole przycisk pozwala pobrać z serwera dokument PDF, zapisać go na urządzeniu oraz otworzyć w jednej z obsługujących do aplikacji.

6.7 Widok zdarzeń finansowych



(a) Widok historii finansów

(b) Widok szczegółów zdarzenia finansowego

Rysunek 6.7: Widoki dotyczące finansów

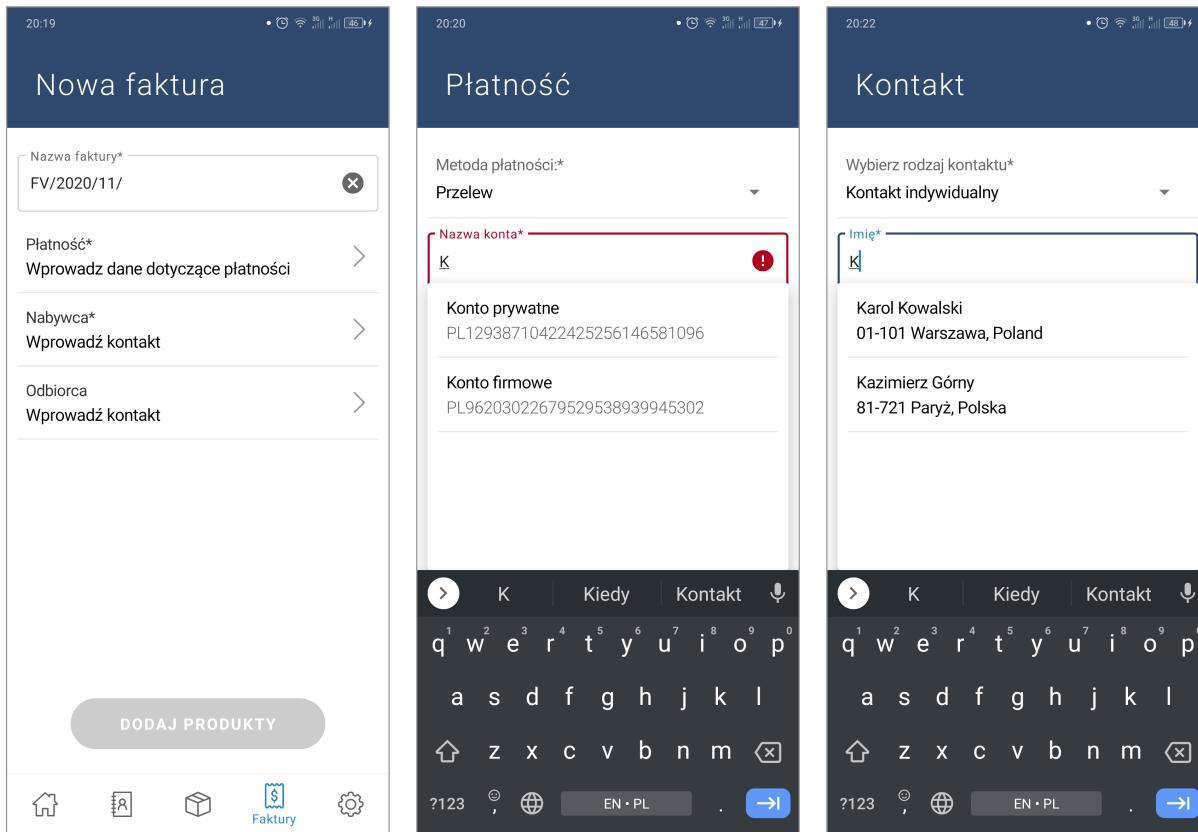
Rysunek 6.7a przedstawia listę zdarzeń finansowych posortowanych według daty wystąpienia. Każdy wiersz jest kompaktową informacją zawierającą tytuł, datę oraz kwotę zmiany. Pozycje będące przychodem lub rozchodem są rozróżniane poprzez zastosowanie różnych kolorów, są nimi odpowiednio kolor zielony i czerwony. Widoczne wiersze zatytułowane numerami faktur są automatycznie wygenerowanymi zapisami, których data wskazuje na czas, w którym użytkownik oznaczył fakturę jako opłaconą. Przycisk umieszczony na dole strony pozwala na dodanie nowego zapisu w rejestrze.



Modyfikacja konkretnych pozycji odbywa się z wykorzystaniem formularza widocznego na rysunku 6.7b. Pola w tym typie widoków są weryfikowane przez dedykowany im validator, a nazwy tych z nich, które są wymagane są udekorowane symbolem gwiazdki. W przypadku wartości tekstowych pola zostały skonfigurowane w ten sposób, aby wyświetlnikowi klawiatura była adekwatna do rodzaju wprowadzanych danych. Skutkiem takiej implementacji jest m.in. wyświetlenie klawiatury numerycznej podczas wprowadzania kwoty dotyczącej zdarzenia finansowego.

Mechanizm dotyczący informowania użytkownika o nieprawidłowościach w wprowadzonych danych, blokowania przycisku oraz sposób opisywania pól przedstawiony w rozdziale 6.3 mają zastosowanie w wszystkich formularzach znajdujących się w aplikacji. Dwa pierwsze pola zachowują się identycznie do tych tam opisanych. Ostatni jest również bardzo zbliżony jednak charakteryzuje go możliwość wprowadzenia wieloliniowych wartości. Komponenty daty i czasu są przyciskami, które otwierają specjalne okna dialogowe zawierające graficzne selektory, co sprawia, że wprowadzanie tych wartości jest wygodniejsze. Więcej informacji na temat nietekstowych metod wprowadzania danych znajduje się w rozdziale 6.10. Na górze znajduje się znana z poprzednich rysunków ikona kosza pozwalająca usunąć zdarzenie z historii finansowej firmy.

6.8 Generowanie faktur



Rysunek 6.8: Widoki pozwalające wprowadzić podstawowe dane dotyczące faktury

Ze względu na ilość potrzebnych danych, cały proces został rozbity na kilka widoków. Pierwsza ich część (Rysunek 6.8) odpowiada za uzupełnienie podstawowych informacji dotyczących faktury, takich jak jej numer, dane nabywcy, odbiorcy oraz dane dotyczące płatności.

Pierwszy ekran na rysunku jest głównym dla tego etapu. Pole dotyczące numeru faktury, domyślnie jest wypełnione prefiksem podanym podczas tworzenia firmy, który również można zmienić modyfikując jej dane. Następne trzy pola są przyciskami, które nawigują do jednego z dwóch kolejnych ekranów na rysunku.

W sytuacji, gdy użytkownik uzupełni dane dotyczące jednego z nich, komunikat z prośbą o wprowadzenie danych zostaje zastąpiony informacją o nich w kompaktowej formie. Uzupełnienie wszystkich wymaganych danych spowoduje odblokowanie przycisku znajdującego się na dole widoku, który pozwoli przejść do kolejnego etapu.

Ekran dotyczący płatności wymaga ustalenia jej terminu. Poza nią, użytkownik ma do wyboru opcję płatności gotówką lub przelewem. W drugim przypadku pojawiają się dwa dodatkowe pola, nazwy i numeru konta. Użytkownik, który zapisał w systemie przynajmniej jeden, może go w tym miejscu wykorzystać, korzystając z sugestii wyświetlnych podczas wprowadzania nazwy.

Formularz dotyczący kontaktu można wypełnić ręcznie lub również skorzystać z wyświetlanych sugestii, które automatycznie wypełnią wymagane pola. Propozycje kontaktów różnią się w zależności od wybranego w pierwszym wierszu typu. To rozróżnienie jest wymagane ze względu na rozbieżność danych, które są zamieszczane na dokumencie.

The image displays two screenshots of a mobile application interface.
(a) Pozycje (Positions): This screen shows a summary of items. It includes a header 'Podsumowanie', a count of 'Ilość pozycji: 2', and a total 'Wartość brutto: 4,499.98'. Below this, there are two items listed: 'Olympus M-5 Mark II Body' with a price of '3,899.99 PLN' and 'M.ZUIKO ED 60mm f/2.8 (czarny)' with a price of '599.99 PLN'. Each item has a red delete icon to its right. At the bottom, there is a blue button labeled 'GENERUJ FAKTURĘ' and a navigation bar with icons for home, document, cube, dollar sign, and settings.
(b) Nowa faktura (New invoice): This screen is for creating a new invoice. It starts with a dropdown for 'Typ*' (Type) and 'Produkt' (Product), followed by fields for 'Nazwa produktu*' (Product name) containing 'M.ZUIKO ED 60mm f/2.8 (czarny)', 'Jednostka miary*' (Unit of measurement) containing 'szt.', and 'Ilość*' (Quantity) with a value of '1'. Below these are fields for 'Cena netto*' (Net price) with a value of '487.8', 'Stawka vat*' (VAT rate) with a value of '23', and 'Cena brutto' (Gross price) with a value of '599.99'. At the bottom is a blue button labeled 'ZAPISZ' (Save) and a navigation bar with icons for home, document, cube, dollar sign, and settings.

(a) Widok przedstawiający listę pozycji

(b) Widok dodawania lub edycji pozycji na fakturze

Rysunek 6.9: Widoki umożliwiające deklarację pozycji faktury

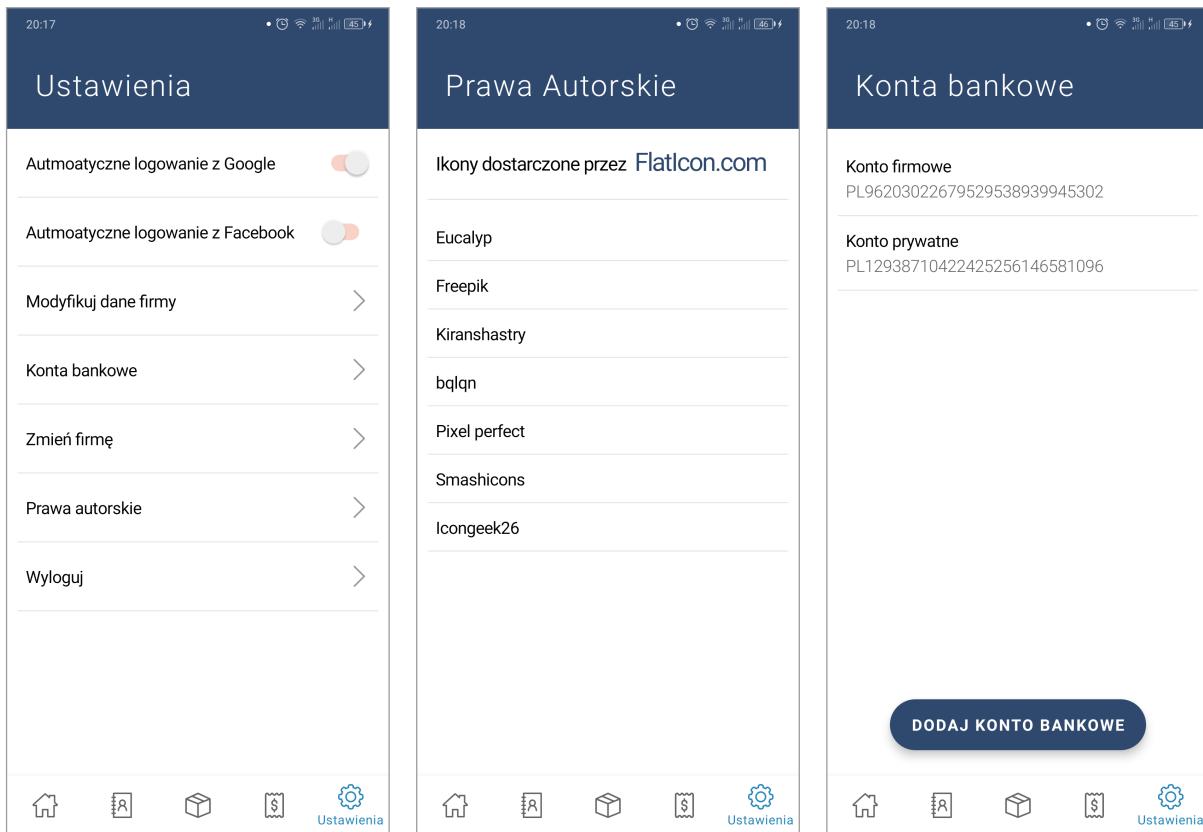
Zadaniem drugiej części interfejsu jest umożliwienie wprowadzenia pozycji, które mają znaleźć się na generowanej fakturze. Widok zaprezentowany na rysunku 6.9a wyświetla bieżącą konfigurację. Na górze ekranu znajduje się komponent zatytułowany jako podsumowanie. Jego zadaniem jest zliczenie oraz wyświetlenie liczby pozycji oraz bieżącej wartości brutto całej faktury. Ze względu na sposób obliczania podatku, suma wartości na liście pozycji może się różnić od prezentowanej wartości. Obok znajduje się przycisk, który nawiązuje użytkownika do ekranu, w którym będzie miał możliwość dodania nowej pozycji. Widoczna poniżej komponentu lista, składa się z wierszy. Każdy z nich zawiera informację o nazwie, cenie brutto i ilości wyrażonej za pomocą przypisanej jednostki miary. Znajdująca się po prawej stronie czerwona ikona pozwala na usunięcie pozycji, a kliknięcie w pozostałą część wiersza spowoduje przekierowanie do widoku edycji zaprezentowanego na rysunku 6.9b. Kliknięcie w znajdujący się na dole przycisk **generuj fakturę** sprawi, że przycisk zostanie zastąpiony animacją informującą o trwającym procesie. Jednocześnie na serwer zostanie wysłane żądanie sieciowe, gdzie zostanie wygenerowany dokument PDF oraz obiekt go reprezentujący. Po pomyślnie zakończonym procesie, użytkownik zostanie przekierowany do detali faktury (Rysunek 6.6b). W przypadku



niepowodzenia zostanie wyświetlona stosowna informacja, a przycisk będzie ponownie dostępny umożliwiając ponowną próbę.

Procesy modyfikacji oraz dodania nowej pozycji do faktury, odbywają się z wykorzystaniem tego samego formularza. Selektor typu pozycji, sprawia, że w zależności od wyboru, formularz udostępnia skaner kodów kreskowych lub nie. W trakcie wypełniania nazwy, użytkownik ma do dyspozycji listę sugestii, gdzie znajdzie produkty lub usługi zapisane w aplikacji. Wybór jednej z nich sprawi, że formularz uzupełni się automatycznie, z wyjątkiem pola dotyczącego ilości. Jednocześnie użytkownik ciągle ma możliwość modyfikacji wybranego modelu, na przykład jego ceny, która może się różnić od sugerowanej. Potwierdzenie danych skutkuje powrotem do widoku z listą pozycji.

6.9 Widok ustawień i praw autorskich

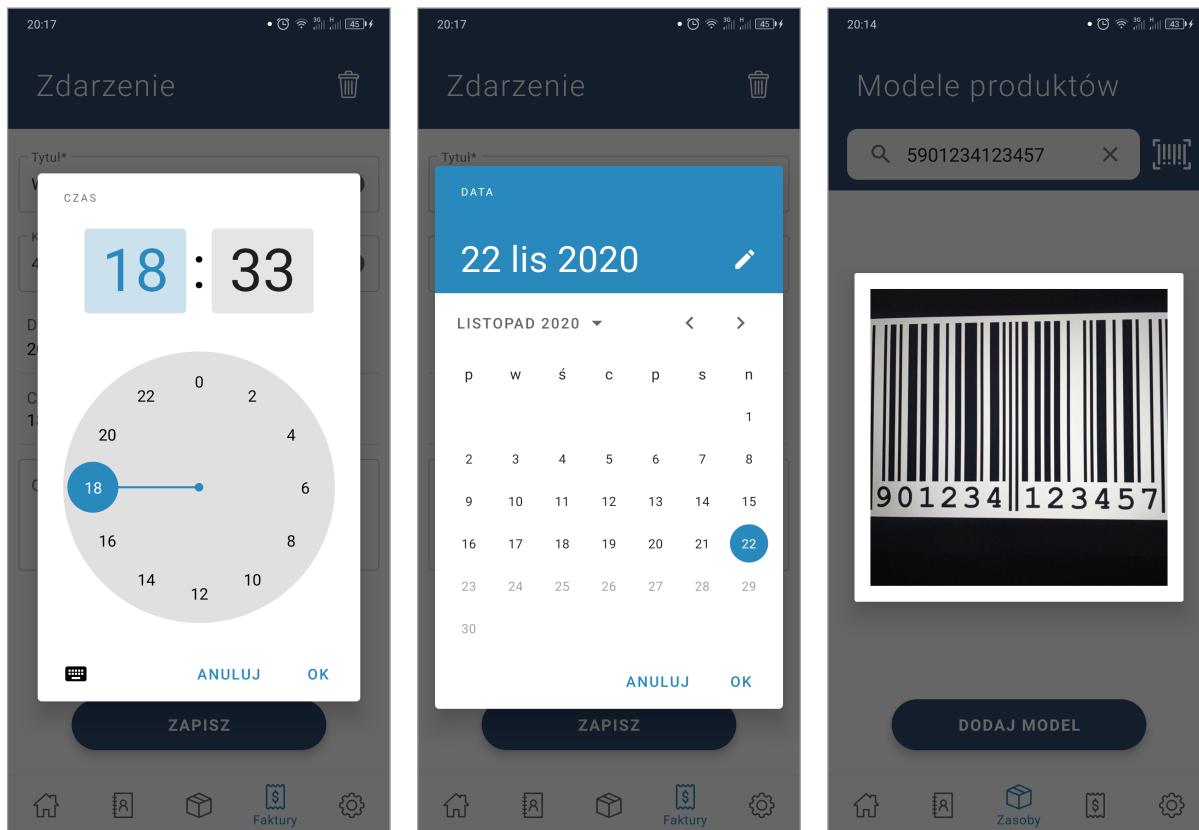


Rysunek 6.10: Widoki dostępne w menu Ustawienia

Ostatni przycisk dostępny w menu daje użytkownikowi dostęp do interfejsu ustawień aplikacji (Rysunek 6.10). Widoczne na głównym ekranie dwa suwaki odpowiadają za aktywację lub dezaktywację mechanizmu, dzięki któremu użytkownik zostaje automatycznie zalogowany do aplikacji zaraz po jej uruchomieniu. Są one ze sobą powiązane, ponieważ obie nie mogą zostać aktywowane jednocześnie. Próba włączenia drugiej opcji, spowoduje wyłączenie pierwszej. Następna pozycja pozwala przejść do formularza, w którym użytkownik ma możliwość modyfikacji danych dotyczących firmy, są to między innymi adres siedziby, numer identyfikacji podatkowej czy też zapisany prefiks numerów faktur. Przycisk dotyczący kont bankowych uruchamia interfejs widoczny na rysunku, dzięki któremu może nimi zarządzać. Wprowadzenie co najmniej jednego, umożliwia aplikacji wsparcie użytkownika w trakcie tworzenia nowej faktury. Weryfikacja poprawności numeru IBAN polega na rozpoznaniu kodu kraju (dwóch pierwszych liter), a następnie w zależności od niego zweryfikowanie dalszej części wprowadzonego numeru. Te różnią się nie tylko długością, ale też

znakami które na danej pozycji mogą występować. Dla kodu wskazującego na Polskę jest to ciąg samych cyfr, jednak na przykład dla kodu włoskiego ten numer składa się kolejno z dwóch cyfr, litery, dziesięciu cyfr oraz dwunastu cyfr lub liter. Wszystkie te aspekty są weryfikowane. Kolejny przycisk umożliwia przełączanie się pomiędzy prowadzonymi na koncie firmami. Interfejs jemu towarzyszący jest podobny do tego opisanego w rozdziale 6.2. Wykorzystane w aplikacji mobilnej ikony, zostały udostępnione nieodpłatnie, pod warunkiem zamieszczenia informacji o ich twórcach. W tym celu został stworzony specjalny widok zatytułowany **Prawa autorskie**, gdzie zamieszczono informację o źródle obrazów oraz interaktywne wiersze z nazwami autorów. Pod każdym z nich znajduje się akcja prowadząca do otwarcia w przeglądarce strony internetowej twórcy. Wybranie ostatniej pozycji w menu spowoduje, że użytkownik zostanie wylogowany z aplikacji i przekierowany do ekranu logowania, gdzie będzie miał możliwość zalogowania się na inne konto.

6.10 Nietekstowe metody wprowadzania danych



Rysunek 6.11: Nietekstowe metody wprowadzania danych

Zadaniem przedstawionych na rysunku 6.11 komponentów jest wsparcie użytkownika, podczas wprowadzania danych. Dzięki nim wprowadzanie danych jest szybsze. Są one bardziej ergonomiczne i sprawiają lepsze wrażenia wizualne w porównaniu do tradycyjnych pól tekstowych.

Wykorzystanie komponentów daty i czasu, wymaga weryfikacji i odpowiedniej obsługi typu zwracanych danych. Podczas ich wdrażania, szczególną uwagę zwrócono na problem różnic stref czasowych i poprawnej konwersji danych do formatów jednoznacznych. Zadbanie o tą kwestię rozwiązuje problemy z spójnością danych w aplikacji klienta i na serwerze oraz sprawia, że przemieszczanie się użytkownika pomiędzy strefami czasowymi zostaje uwzględnione. Oba komponenty pochodzą z biblioteki Google Material dla systemu Android. Czas wprowadzany jest przy użyciu dwóch tarcz wyświetlanych po sobie, które pozwalają na wybór kolejno godziny oraz minut. Data wprowadzana jest przy pomocy widocznego na rysunku kalendarza.



W zależności od zastosowania istnieje możliwość ograniczenia możliwości wyboru. Taka sytuacja zachodzi w przypadku określania terminu płatności faktury, gdzie użytkownik nie ma możliwości wyboru daty, która już minęła.

Skaner kodów kreskowych wykorzystuje bibliotekę Google Mobile Vision, która udostępnia detektor analizujący obrazy w poszukiwaniu kodów kreskowych. Ze względu na wymagany dostęp do kamery zaimplementowano mechanizm, który weryfikuje i wnioskuje o odpowiednie zgody użytkownika. Jeżeli są one udzielone kamera rozpoczyna rejestrowanie obrazu i detekcję kodów. Procesowi towarzyszy zaimplementowane okno dialogowe wyświetlające podgląd. Skaner wspiera kody jednowymiarowe: EAN_8, EAN_13, UPC_A, UPC_E oraz ITF.

Konfiguracja i uruchomienie systemu

W tym rozdziale przedstawiono kroki, które należy wykonać, aby skonfigurować i uruchomić system. Ze względu na konieczność wykonania konfiguracji usług Google oraz Facebook, procedura składa się z wielu etapów. Do jej przeprowadzenia wykorzystywane są prywatne dane, które nie powinny być dostępne publicznie i z tego względu nie zostały zawarte na nośniku.

7.1 Środowisko

Konfiguracja środowiska, które wykorzystano do napisania poniższego opisu prezentuje się następująco:

- System operacyjny: Ubuntu 20.04.1 LTS
- Oprogramowanie Docker 19.03.12
- Platforma Android Studio 4.1.1

Do wykonania kolejnych kroków potrzebne będą dane dotyczące systemu. Ich zbiór zamieszczono poniżej.

- Nazwa pakietu aplikacji mobilnej: dev.szafraniak.bm_mobileapp
- Nazwa domyślnej aktywności: dev.szafraniak.bm_mobileapp.presentation.login.LoginActivity_
- Dane konta administratora w oprogramowaniu Keycloak: Login: admin Hasło: admin
- Adres serwera Keycloak: <http://localhost:8080/auth>
- Adres konsoli Facebook: <https://developers.facebook.com>
- Adres konsoli Google: <https://console.developers.google.com>

7.2 Konfiguracja usług zewnętrznych

W tej części konfiguracji opisano kroki, które należy wykonać wykorzystując interfejsy zewnętrzne. Do konfiguracji usług Google oraz Facebook konieczne jest wygenerowanie podpisu cyfrowego aplikacji mobilnej. W tym celu należy wykonać poniższe kroki:

1. Z załączonego do pracy nośnika przenieś katalogi BM-Servers oraz BM-MobileApp na dysk komputera.
2. Uruchom terminal oraz wykorzystując zainstalowany wraz z platformą Android Studio program keytool wykonaj poniższe polecenie i uzupełnij wymagane dane:
`keytool -genkey -v -keystore BmMobileApp.jks -alias BmMobileApp -keyalg RSA
-keysize 2048 -validity 10000`
3. Utworzony plik BmMobileApp.jks przenieś do katalogu
BM-MobileApp/certificate/
4. Otwórz projekt BM-MobileApp w środowisku Android Studio.



5. Otwórz plik BM-MobileApp/app/build.gradle.
6. Edytuj sekcję android.signingConfigs.defaultConf zmieniając jej wartości keyPassword i storePassword na hasło podane podczas generowania pliku BmMobileApp.jks.
7. Korzystając z Android Studio, należy wybrać zestaw narzędzi Gradle i uruchomić zadanie: BM-MobileApp → Tasks → android → signingReport.
8. W wyświetlonej treści należy odszukać wartość SHA1 oraz ją zachować.

Posiadając powyższe dane należy wykonać konfigurację usług zewnętrznych Google oraz Facebook, która odbywa się na portalach tych producentów.

Google:

1. Załóż konto oraz zaloguj się w konsoli Google.
2. Wybierz przycisk wybierz projekt, a następnie nowy projekt.
3. Nadaj mu dowolną nazwę i zaakceptuj dane.
4. Wybierz stworzony projekt.
5. Przejdz do podglądu interfejsów API.
6. Przejdz do zakładki ekran akceptacji.
7. Zaznacz zewnętrzny typ użytkownika oraz potwierdź wybór.
8. Wprowadź nazwę aplikacji Business-manager i dwukrotnie podaj adres kontaktowy email w wymagane pola.
9. Przejdz do końca formularza nic nie wypełniając i wcisnij przycisk powrót do panelu.
10. Wciśnij przycisk opublikuj aplikację i potwierdź akcję.
11. Przejdz do zakładki dane logowania.
12. Wybierz opcję utwórz dane logowania i Identyfikator klienta OAuth.
13. Wybierz typ aplikacji Andorid, wprowadź dowolną nazwę, nazwę pakietu która znajduje się na początku rozdziału oraz wygenerowany odcisk SHA1.
14. Zachowaj utworzony identyfikator klienta Android.
15. Ponownie wybierz utwórz dane logowania i Identyfikator klienta OAuth.
16. Wybierz typ Aplikacja Internetowa, wprowadź dowolną nazwę oraz zaakceptuj formularz.
17. Zachowaj identyfikator klienta (Client Id) oraz jego sekret (Client Secret).

Facebook:

Do konfiguracji usługi potrzebny jest wcześniej wygenerowany podpis SHA1 przedstawiony w formacie Base64. Należy zwrócić uwagę, że występujące w SHA1 dwukropki nie są częścią szesnastkowego kodu. Konwersję można przeprowadzić wykorzystując dowolny kontener.

1. Załóż konto i zaloguj się w konsoli Google.
2. Wybierz przycisk Get Started i przejdź przez kolejne kroki.
3. Dodaj aplikację wybierając kolejno create app oraz Build Connected Experience.
4. Wprowadź dowolną nazwę projektu i przejdź dalej.



5. Wybieramy przycisk **set up** znajdujący się obok pozycji **Facebook Login**.
6. Wybierz typ aplikacji **Android**
7. Pomiń dwa pierwsze kroki i przejdź do kroku 3. **Tell Us about Your Android Project.**
8. Wprowadź wymagane informacje. Dane znajdują się na początku rozdziału.
9. Zapisz zmiany i przejdź do kolejnego kroku.
10. Na dole w polu **Key Hashes** wprowadź podpis **SHA1** w formacie **Base64**.
11. Zapisz zmiany i pomiń resztę kroków.
12. Przejdz do zakładki **settings** oraz wybierz **Basic**.
13. Zachowaj widoczny na ekranie identyfikator aplikacji (App ID) oraz sekret (App Secret)

Na górze ekranu znajduje się suwak pozwalający przełączać usługi pomiędzy trybami pracy: produkcyjnym i rozwojowym. Od niego zależy, którzy użytkownicy będą mieli możliwość zalogowania się do aplikacji. Tryb produkcyjny wymaga podania kilku dodatkowych informacji i pozwala dowolnemu użytkownikowi wykorzystać konto Facebook do zalogowania się w realizowanym systemie. Tryb rozwojowy na tą czynność pozwala jedynie zadeklarowanym testerom, właścicielom i programistom. Aby takowych zadeklarować należy przejść do zakładki **Roles** i tam zdefiniować użytkowników. Co ważne, aby zadeklarowany użytkownik mógł korzystać z usługi, koniecznie musi zaakceptować „zaproszenie”. Musi zalogować się do konsoli Google kliknąć na ikonę użytkownika i wybrać **Requests**. W tym miejscu może zaakceptować zaproszenie. Póki tego nie zrobi w karcie **Roles** przy jego koncie będzie wyświetlana informacja **Pending**.

7.3 Konfiguracja systemu

Następnym etapem konfiguracji jest wprowadzenie uzyskanych danych do oprogramowania Keycloak.

1. Otwórz katalog **BM-Servers**.
2. Uruchom w nim terminal.
3. Upewnij się, że korzystasz z konta z prawami administratora. (Ewentualnie przełącz się na nie wykonując polecenie **sudo su**).
4. Wykonaj polecenie **docker-compose build**.
5. Uruchom kontener wykonując polecenie **docker-compose up**.
6. Przejdz do interfejsu oprogramowania Keycloak.
7. Wybierz **Administration Console**
8. Zaloguj się wykorzystując dane znajdujące się na początku rozdziału.
9. Upewnij się ze wybrany projekt to **Business-manager**.
10. Przejdz do zakładki **Identity Providers**.
11. Edytuj pozycję **Google** wprowadzając identyfikator klienta oraz sekret uzyskany w procesie konfiguracji usługi Google dla aplikacji internetowej.
12. Edytuj pozycję **facebook** wprowadzając identyfikator aplikacji oraz sekret uzyskany w procesie konfiguracji usługi Facebook.
13. Wyloguj się.



Kolejnym etapem jest skonfigurowanie aplikacji mobilnej do pracy z usługami zewnętrznymi.

1. Powróć do platformy Android Studio.
2. Otwórz plik `BM-MobileApp/app/build.gradle`.
3. Zamień wartość przypisaną do pola `GOOGLE_CLIENT_ID` na otrzymany w procesie konfiguracji usługi Google dla aplikacji internetowej identyfikator klienta (Client Id). Ważne wartość musi być umieszczone w apostrofach i cudzysłowie.
4. Zamień wartość przypisaną do pola `RESOURCE_SERVER_URL` na adres, gdzie - z poziomu zainstalowanej na urządzeniu aplikacji mobilnej – będzie znajdował się kontener z częścią serwerową systemu. Wykorzystaj do tego port 8081 pod którym w kontenerze udostępniany jest serwer REST API. Ważne, wartość musi być umieszczone w apostrofach i cudzysłowie.
5. Otwórz plik `BM-MobileApp/app/src/main/res/values/strings.xml` oraz zamień wartości przypisane do pól `facebook_app_id` oraz `fb_login_protocol_scheme` na odpowiednio: identyfikator aplikacji uzyskany w procesie konfiguracji usługi Facebook oraz ten sam identyfikator poprzedzony tekstem `fb`. Dla przykładowego identyfikatora `1234567890` wartość pierwszego z pól to `1234567890`, a drugiego `fb1234567890`.

7.4 Instalacja i uruchomienie

Aby umożliwić użytkownikowi korzystanie z systemu należy uruchomić część serwerową, jeżeli przeprowadzono konfigurację opisaną w wcześniejszych krokach, serwer jest już uruchomiony. W sytuacji, gdy istnieje już skonfigurowany kontener jednak jest wyłączony należy go uruchomić wykonując polecenie `docker-compose up`.

Instalacja aplikacji mobilnej wymaga wcześniejszego wygenerowania instalowanego pliku z rozszerzeniem APK. Należy to wykonać w następujący sposób.

1. Powróć do platformy Android Studio.
2. Korzystając z narzędzia Gradle należy uruchomić zadanie:
`BM-MobileApp → Tasks → other → packageRelease`.
3. Przejdź do katalogu `BM-MobileApp/app/build/outputs/apk/release` i przenieś wygenerowany plik `app-release.apk` na urządzenie z systemem Android.
4. Zainstaluj aplikację na urządzeniu wykorzystując wgrany plik APK.

Podsumowanie

Celem niniejszej pracy było zaprojektowanie i zrealizowanie oprogramowania wspomagającego funkcjonowanie małych działalności gospodarczych. System składający się z aplikacji mobilnej na platformę Android oraz zespołu narzędzi tworzących warstwę serwerową, realizuje wszystkie postawione we wstępie wymagania funkcjonalne i niefunkcjonalne. Analiza statystyk dotyczących udziału poszczególnych wersji systemu Android na rynku, pozwoliła osiągnąć dobry kompromis pomiędzy wykorzystaniem nowoczesnych technologii oraz zapewnieniem wsparcia dla jak największej liczby urządzeń. Wykonany system pozwala generować faktyry w formacie PDF, prowadzić spis produktów w magazynach oraz rejestr zdarzeń finansowych. Umożliwia wprowadzanie kontaktów i obsługę numerów telefonicznych. Interfejs graficzny udostępnia dwa języki, daje możliwość wykorzystania kamery urządzenia do skanowania kodów kreskowych oraz posiada mechanizmy wsparcia użytkownika w wyszukiwaniu treści oraz uzupełnianiu formularzy. W ramach pracy dyplomowej znaleziono, skonfigurowano i zaimplementowano odpowiednie rozwiązania, aby umożliwić użytkownikowi logowanie do systemu za pośrednictwem usług zewnętrznych firm Google oraz Facebook. Przedstawiane w aplikacji informacje są zawsze aktualne. Synchronizacja z serwerem sprawia, że dane użytkownika na każdym urządzeniu są takie same.

Kierunkiem w którym system mógłby się rozwijać jest wprowadzenie mechanizmu zarządzania firmą przez więcej niż jednego użytkownika. W takim modelu korzystne byłoby również zaprojektowanie ról, dzięki którym zostałyby określone prawa użytkownika do wykonywania danych operacji. Nadanie dostępu kilku użytkownikom do wybranych funkcjonalności takich jak prowadzenie spisu produktów w magazynach, wystawianie faktur czy rejestrowanie zdarzeń finansowych sprawia, że system staje się przyjazny nie tylko dla kilkuosobowych firm, ale również dla tych średniej wielkości.



Bibliografia

- [1] Android 8.0 oreo. Strony internetowe: <https://developer.android.com/about/versions/oreo>.
- [2] Dagger. Strony internetowe: <https://dagger.dev>.
- [3] Docker. Strony internetowe: <https://docs.docker.com>.
- [4] Git. Strony internetowe: <https://git-scm.com/book/en/v2>.
- [5] Google mobile vision. Strony internetowe: <https://developers.google.com/vision>.
- [6] Gson. Strony internetowe: <https://github.com/google/gson>.
- [7] itext. Strony internetowe: <https://itextpdf.com/en>.
- [8] Keycloak. Strony internetowe: <https://www.keycloak.org>.
- [9] Kod źródłowy systemu android. Strony internetowe: <https://source.android.com/setup/build/downloading>.
- [10] Lombok. Strony internetowe: <https://projectlombok.org>.
- [11] Material design. Strony internetowe: <https://material.io/develop/android>.
- [12] Mysql. Strony internetowe: <https://dev.mysql.com>.
- [13] ReactiveX. Strony internetowe: <https://reactivex.io>.
- [14] Retrofit. Strony internetowe: <https://square.github.io/retrofit>.
- [15] RxLifecycle. Strony internetowe: <https://github.com/trello/RxLifecycle>.
- [16] Spring. Strony internetowe: <https://docs.spring.io/spring-framework/docs/current/reference/html>.
- [17] Udział mobilnych systemów operacyjnych na polskim rynku. Strony internetowe: <https://gs.statcounter.com/os-market-share/mobile/poland/#monthly-202011-202011-bar>.
- [18] Udział poszczególnych wersji systemu android na polskim rynku. Strony internetowe: <https://gs.statcounter.com/android-version-market-share/mobile-tablet/poland/#monthly-202011-202011-bar>.



Zawartość płyty CD

Na załączonym do pracy nośniku znajduje się kopia niniejszej pracy dyplomowej oraz dwa katalogi. Pierwszy z nich zawiera pliki źródłowe aplikacji mobilnej, natomiast w drugim znajduje się kod serwera REST, a także konfiguracje bazy danych i oprogramowania Keyclaok.

