Studia i Materiały Informatyki Stosowanej, Tom 12, Nr 2, 2020
str. 15-21

Mobilne zarządzanie wynajmowaniem mieszkań. Aplikacja na platformę Android z wykorzystaniem Node.js oraz MySQL

Belco Sangho, Zbyszko Królikowski

Instytut Informatyki
Uniwersytet Kazimierza Wielkiego w Bydgoszczy

Streszczenie: *Urządzenia mobilne pozwalają na kreowanie aplikacji opierających się na ciągłej dostępności do Internetu i użytkownika, umożliwiając wprowadzenie na nowy poziom wcześniej istniejących koncepcji. Celem tego badania jest program, który umożliwi zarządzanie najmem mieszkań, automatyzację opłat oraz komunikację wynajmującego z lokatorami, jednocześnie znajdując się w zasięgu ręki. Zostało to zrealizowane przy pomocy aplikacji mobilnej napisanej w środowisku Java, serwera stworzonego przy pomocy środowiska Node.js, oraz bazy danych stworzonej w systemie zarządzania MySQL Workbench. Finalny produkt jest ogromnym ułatwieniem w zarządzaniu mieszkaniami, zarówno dla właściciela jak i lokatora.*

Słowa kluczowe: *Android Studio, MySQL Workbench, Node.js, aplikacja moblina, serwer, Java*

Mobile management of flat rental - application on the Android platform using Node.js and MySQL

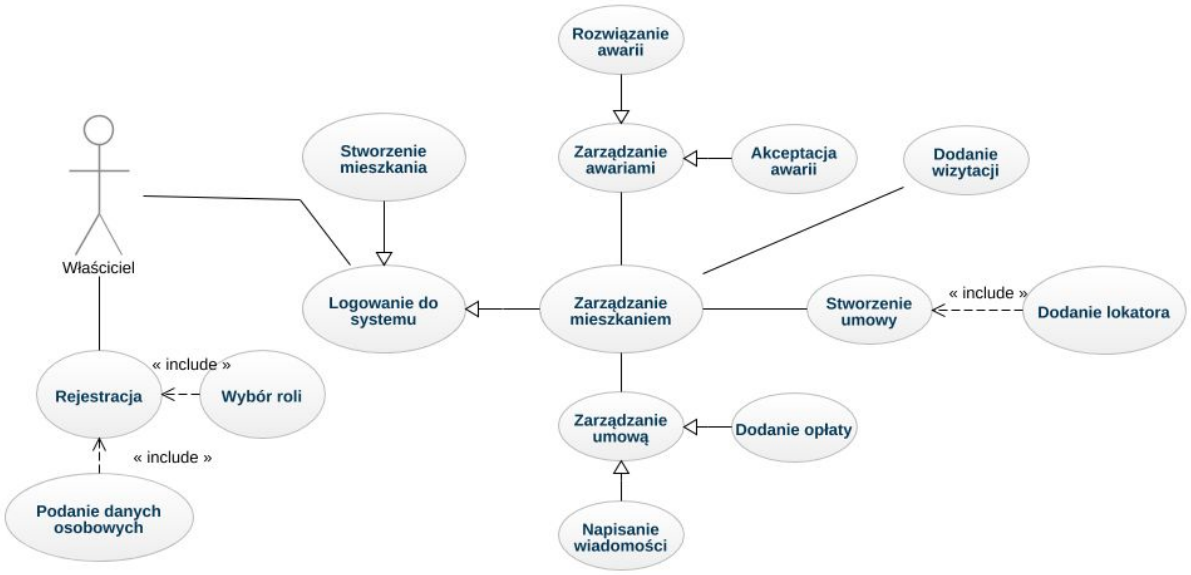
Abstract: *Mobile devices allow us to create applications based on constant availability of internet and a user, which we can use to improve previously existing ideas. The aim of the study is to create program, capable of helping with rental management, automatic fees and communication between owner and tenant, while safely remaining within reach. Such application was created with a usage of environment Android Studio with help of Java language, while the backend server was made with a JavaScrip runtime environment Node.js, and a database formed in relation database management system MySQL Workbench. Final product is an enormous help with rental management, both for owner and tenant.*

Keywords: *Android Studio, MySQL Workbench, Node.js, mobile application, server, Java*

1 Wprowadzenie

W związku z ogromnym wzrostem cen mieszkań na przestrzeni ostatnich pięćdziesięciu lat[1], popularnym źródłem zarobku wielu dużych i małych firm stało się wynajmowanie przestrzeni do życia dla rodzin. Na przeciw potrzebom takich firm wyszło wiele oprogramowań do zarządzania nieruchomościami, pozwalającymi sprawnie zawiązywać umowy, pilnować płatności oraz wyposażenia. Pozostawiło to jednak małe firmy i osoby prywatne, nierzadko posiadające tylko kilka mieszkań, w sytuacji, gdzie zakup i utrzymanie takich aplikacji byłyby nieopłacalne. Często też oprogramowanie to nie wspierało ciągle rozwijającego się rynku urządzeń mobilnych, który ułatwia monitorowanie poprzez prostotę i szybkość dostępu do informacji.

Idea stworzenia aplikacji która wyszłaby naprzeciw aktualnym potrzebom powstała podczas zajęć na Uniwersytecie Kazimierza Wielkiego w grupie współpracującej z firmą Logon. Finalnie została ona dokończona jako praca inżynierska. Projekt



Rysunek 1: Diagram przypadków użycia 1

miał powstać przy pomocy najnowszych technologi, aby zapewnić mu zarówno prostotę obsługi, oraz prędkość działania.

Jak wynika z raportów popularności systemów operacyjnych zainstalowanych na urządzeniach mobilnych, ponad połowa rynku należy do Androida, jednak poszczególne raporty różnią się pod względem dokładnej wartości. Z tego też powodu zdecydowano się na wykorzystanie tej właśnie platformy. O popularności tego systemu zadecydowała głównie otwartość oprogramowania, która pod licencją Apache zezwala na dowolną dystrybucję i modyfikowanie jego kodu źródłowego, oraz możliwość wykorzystywania go za darmo. Pozwala to na dostosowanie systemu do wymagań twórcy urządzenia.

Aplikację serwerową zdecydowano się napisać w Node.js, który oferuje nam możliwość napisania oprogramowania bazującego na nieskończonej pętli zdarzeń, odpowiadającej na przesyłane tam zapytania. Środowisko to działa bardzo dobrze w trybie asynchronicznym, co pozwala na maksymalne wykorzystanie dostępnych nam zasobów serwera.

Częścią wymaganą w takim rozwiązaniu jest baza danych, która została stworzona w języku MySql. Relacyjne bazy danych charakteryzują się prostotą w tworzeniu oraz interakcji z serwerem, co było powodem takiego wyboru.

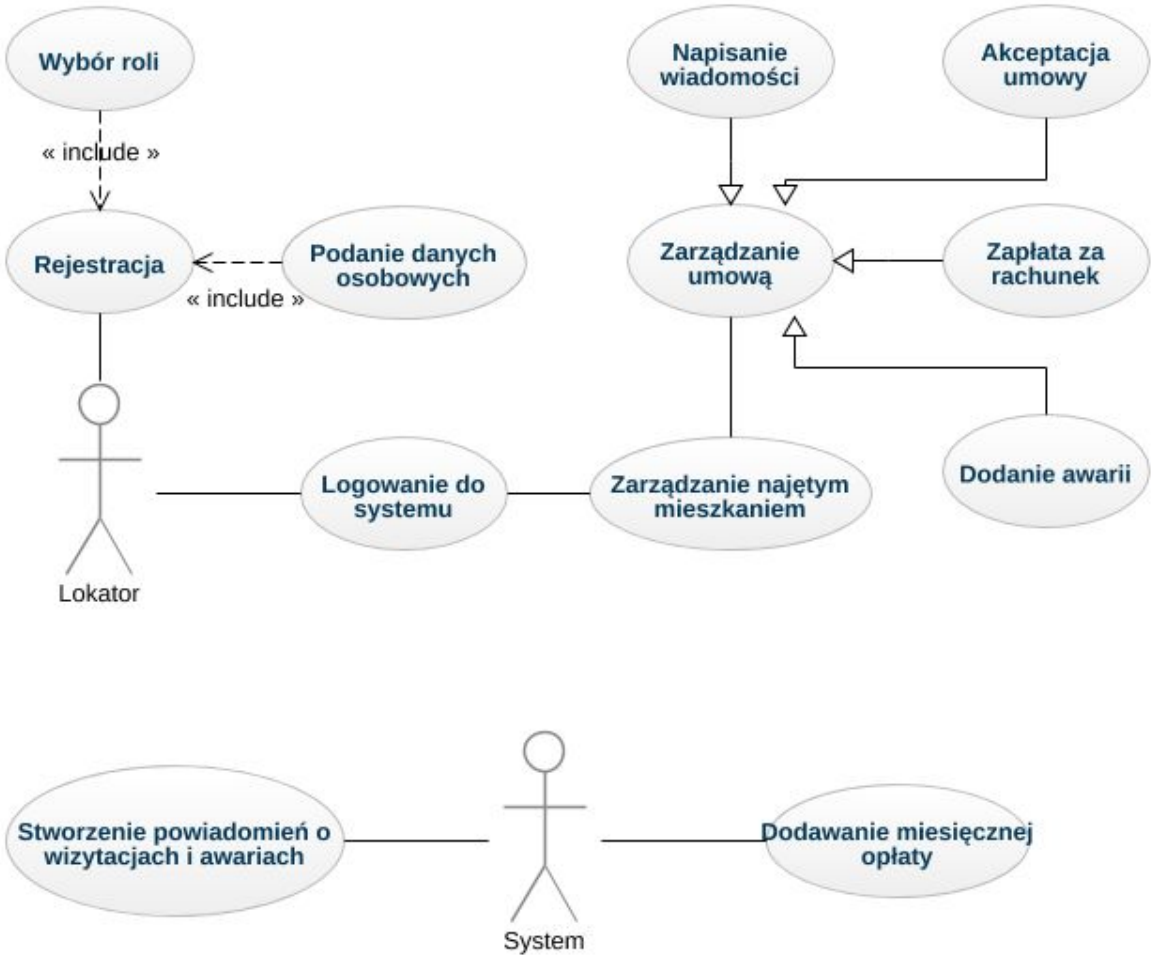
2 Metody

2.1 Diagram przypadków użycia

Aby stworzyć tak duży system, pierwszym krokiem powinno być określenie jego ram. Diagram przypadków użycia (Rys. 1 i 2) jest graficznym przedstawieniem wielkości i funkcjonalności architektury, dając wszystkim jasną wizję tego co należy zrobić[2]. Ważnym jego elementem jest uniezależnienie od implementacji, i pokazanie możliwych akcji (opisanych w elipsach) każdego z aktorów. Pozwala to w dalszym etapie programowania na uniknięcie sprzeczności odnośnie tego co jest wymagane, a co jedynie mogłoby się pojawić w finalnym produkcie. Jako że system jest dość rozległy, wymagania są dość duże. Część funkcjonalności jest jednak taka sama dla właściciela i lokatora.

2.2 MySQL Workbench

Aby stworzyć relacyjną bazę danych, zdecydowano się wykorzystać pogram posiadający interfejs graficzny[3], który przyśpieszył inicjalną prace podczas tworzenia schematu bazy. Zmniejszyło to ilość pisania bezpośredniego kodu MySQL. Środowisko to pozwala też na tworzenie wyzwalaczy, wykorzystywanych przy wykonywaniu akcji na rekordach tabeli. Procedury dają możliwość na wywoływanie operacji na bazie, przy minimalnym podawaniu danych i ekspozycji wewnętrznej struktury przez



Rysunek 2: Diagram przypadków użycia 2

serwer. Za pomocą zdarzeń tworzy się włączające się o specyficznych datach funkcje, co pozwala na zautomatyzowanie pewnych części sytemu, takich jak opłaty za mieszkanie. Widoki wykorzystuje się do łączenia danych z kilku tabel, wybierając interesujące nas informacje, za pomocą kluczy głównych identyfikujących rekord oraz kluczy obcych, które wskazują do którego rekordu należy informacja zawarta w innej tabeli.

2.3 Node.js

Środowisko Node.js zyskało swoją popularność jako część inicjatywy *JavaScript Everywhere*, odnoszącą się do możliwości pisania w tym samym języku aplikacji webowych zarówno po stronie serwera, jak i klienta. Biorąc pod uwagę działanie w otwartym kodzie, każdy może modyfikować i dostosowywać to środowisko do swoich potrzeb[4]. Domyślnym menadżerem paczek jest Npm, a aktywne środowisko programistyczne dostarcza ogromną ilość bibliotek do pobrania. Ważnym aspektem środowiska jest działanie tylko wtedy, kiedy jest wymagana od niego praca, co daje możliwość budowania dużych aplikacji mądrze zarządzającej zasobami. Wraz ze stworzeniem nowych bibliotek istnieje nawet możliwość pisania aplikacji mobilnych w JavaScripcie, aczkolwiek ta opcja nie została tu wykorzystana.

Jako ze Node.js nie posiada domyślnego edytora tekstowego, wykorzystano Visual Studio Code. Środowisko to pozwala na dostosowanie go do każdego języka, oraz szybki dostęp do terminala, z którego pomocą wywoływano komendy Npm-a oraz Noda-

a.

Najważniejszym komponentem wykorzystanym podczas tworzenia serwera jest biblioteka Express. Obecne są w niej ścieżki[5] tworzące interfejs do komunikacji za pomocą zapytań Http. Są one przeszukiwane kolejno od góry do dołu, aż zostanie znaleziona odpowiednia trasa. Dodatkowo wykorzystano kilka pomocniczych bibliotek do komunikacji z bazą danych. Asynchroniczność całego środowiska dodatkowo pozwoliła na szybkie działanie.

Ze względu na powstawanie aplikacji mobilnej i serwera jednocześnie, testowanie ścieżek odbyło się przy pomocy dodatkowego programu Postman, który daje możliwość wysyłania zapytań http i sprawdzania odpowiedzi w każdym formacie, uniezależniając go od programu na urządzeniu mobilnym.

2.4 Android Studio

Zbudowane na otwartej wersji środowiska IntelliJ IDEA, Android Studio łudząco przypomina wyglądem i działaniem program który posłużył za jego podwaliny. Jest on darmowy, oraz posiada oficjalne wsparcie Google, producenta systemu Android. Z tego powodu nie należy się dziwić, że jest on obecnie jednym z najefektywniejszych sposobów pisania aplikacji na tą platformę. Domyślnym językiem programowania jest Java, i to ona została użyta w projekcie, aczkolwiek rosnące wsparcie dla Kotlina może niedługo oznaczać wyparcie pierwotnego języka. Szerokie wsparcie bibliotekami tworzonymi zarówno przez związanych i nie związanych z producentem programistów zostało wykorzystane przy integracji z serwerem.

Ekrany tworzono przy pomocy *Relative View*, który pozwala na pozycjonowanie elementów w zależności od siebie, oraz *Linear View*, który układa elementy jeden po drugim w orientacji poziomej bądź pionowej[6]. Duża część funkcjonalności wykorzystywała *RecyclerView*, pozwalający na proste i optymalne dla pamięci wyświetlenie podobnych do siebie elementów dodawanych dynamicznie.

Do komunikacji z serwerem zastosowano tak zwaną *singelton-ową* klasę, która może posiadać tylko jedną instancję. Chroni nas to przed jednoczesnym istnieniem wielu kolejek zapytań, co negatywnie wpływało by na logikę działa programu, oraz mogłoby potencjalnie doprowadzić do wykorzystania zbyt wielu zasobów, i w efekcie wymuszenie przez system zamknięcia programu.

Menadżerem paczek jest Gradle. Zarządza on pochodzeniem bibliotek ze zdalnych repozytoriów, co znacząco ułatwia ich importowanie, oraz pilnowanie ich wersji.

3 Rezultaty

3.1 Baza danych

Efektem pracy z systemem zarządzania MySQL Workbench jest baza danych pozwalająca na spełnienie wymagań funkcjonalnych (Rys. 3). Posiada ona dziesięć tabel. Przyjęta została logika, aby nazywać po polsku pola własne tabeli, a klucze obce w języku angielskim. Do paru najważniejszych rzeczy należy:

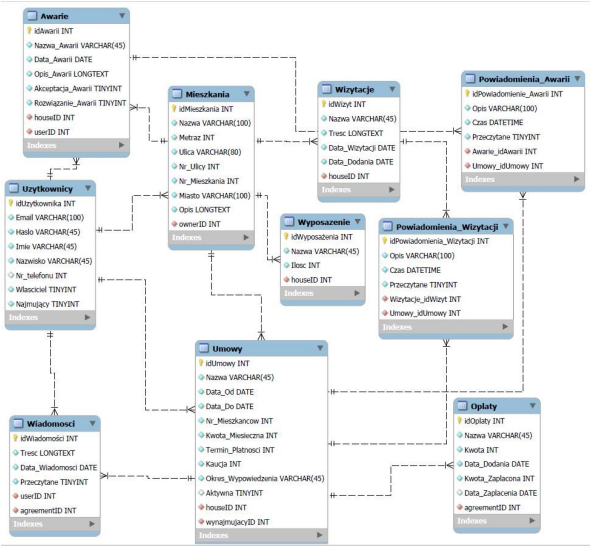
- Wiele pól w tabeli Umowy, pozwalających na automatyczne wygasanie umów, naliczanie opłat do tabeli Opłaty oraz komunikację przy pomocy tabeli Wiadomości, między użytkownikami, których wiąże umowa.
- Generowanie powiadomień o awariach oraz wizytacjach, odpowiednio dla wszystkich lokatorów i właściciela w przypadku awarii, oraz tylko dla lokatorów w razie wizytacji, razem z przypomnieniem na dzień przed planowanym przybyciem posiadacza mieszkania.
- Duża ilość widoków, które są nieobecne na schemacie bazy danych, pozwalających na sprawne przekazywanie informacji do serwera.
- Kilka funkcji, automatycznie włączanych podczas wyświetlania wiadomości, aby przy następnym odpytaniu bazy były one zaznaczone jako przeczytane.

Finalnie baza pozwoliła na sprawne przekazywanie informacji do serwera.

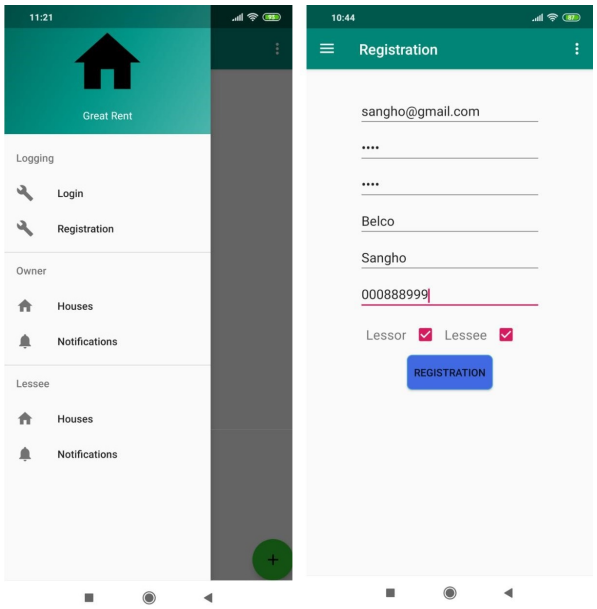
3.2 Serwer

Przy użyciu biblioteki Express udało się utworzyć około 40 zapytań z serwera do bazy danych. Pozwalają one na utworzenie dróg dostępu do informacji, które są później przesyłane w formacie danych JSON do aplikacji mobilnej. Struktura tych danych jest tekstowa, oraz wszechobecna w większości języków programowania, co pozwoliło z kolei na proste odwzorowanie na obiekty w Javie.

Autoryzacja odbywa się poprzez sprawdzanie poprawnych danych podczas logowania, które w poprawnym przypadku pozwalają na dostęp do identyfikatora odpowiedniego użytkownika. Podczas programowania aplikacji upewniono się, że nawet w przypadku komunikacji z drugim użytkownikiem, dane dotyczące haseł(zarówno tego użytkow-



Rysunek 3: Schemat bazy danych



Rysunek 4: Rejestracja i menu

nika jak innych) bądź identyfikatora innego użytkownika nigdy nie dostawały się do programu, co jest dodatkowym źródłem zabezpieczenia.

W celu zapewnienia zdalnego dostępu do serwera, zdecydowano się na wykorzystanie platformy w chmurze Heroku[7]. Zapewnia ona darmową możliwość do stworzenia tego typu małych aplikacji webowych, jeśli nie przekroczymy danej wielkości bazy, oraz pogodzimy się z wolniejszą responsywnością.

3.3 Aplikacja mobilna

Interfejs użytkownika finalnie posiada dwadzieścia ekranów. Jednym z pierwszych ekranów jest rejestracja, gdzie mamy możliwość wybrania roli właściciela bądź lokatora. Istnieje możliwość posiadania obu ról (Rys. 4). Poruszanie się po aplikacji zostało zrealizowane za pomocą menu wysuwanego z lewej krawędzi. Pozwala to na przełączanie się między widokiem mieszkań a powiadomień, oraz na zmianę funkcji z najemcy z wynajmującego i odwrotnie.

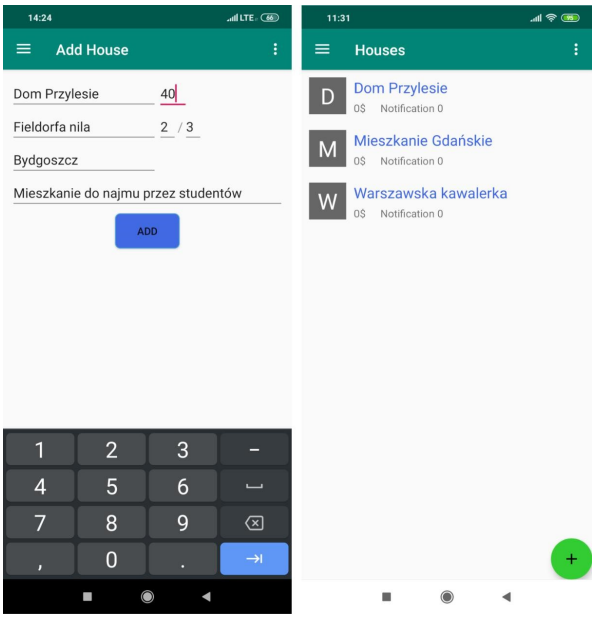
Pierwszy ekran widoczny po zalogowaniu ukazuje nam z perspektywy właściciela przynależne mu mieszkania (Rys. 5), bądź analogiczny widok dla lokatora. Za pomocą przycisku zlokalizowanego w prawym dolnym rogu mamy możliwość dodania nowego lokalu. Funkcjonalność ta powtarza się na każdym ekranie, który pozwala na dodanie nam nowego elementu. Dodatkowo pod nazwą mieszkania widzimy ogólny bilans pieniężny dla wszystkich umów, oraz informacje o wiadomościach od lokatora i nowych awariach. Kolejne widoki to:

- Ekran umów, widoczny jedynie dla właściciela, który pozwala podejrzeć wszystkie aktywne umowy z bilansami spłat rachunków przez lokatorów oraz możliwość wejścia w szczegółowe informacje o opłatach (Rys. 6). Na przyciskach pojawiają się też informacje o tym czy jest obecna nierozwiązana awaria, bądź nowa wiadomość od lokatora.

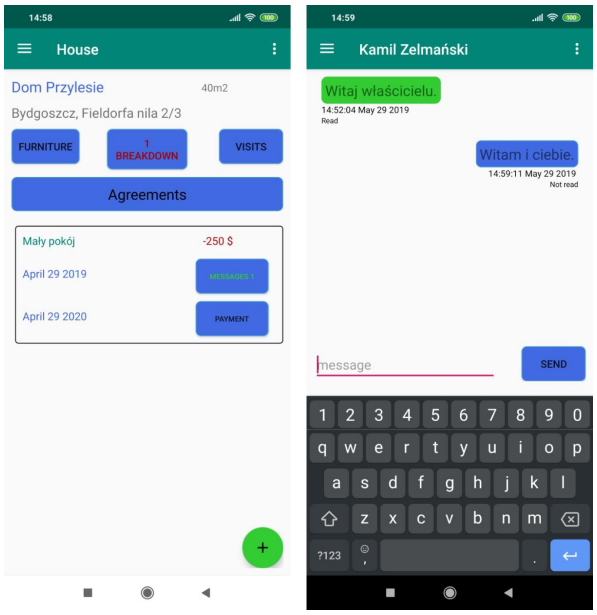
- Komunikator, pozwalający na wymianę wiadomości pomiędzy właścicielem oraz lokatorem (Rys. 6). Pod wiadomościami widać godzinę i datę wysłania, jak i to czy odczytaliśmy ją podczas ostatniej konwersacji.

- Przeglądanie awarii, pozwalające nam na ich akceptowanie (aby przekazać informację o zauważeniu do lokatorów), oraz rozwiązanie kiedy problem zostanie zażegnany (Rys. 8). Dodawanie awarii jest stosunkowo proste, ponieważ data dodaje się automatycznie. Przeglądanie i dodawanie wizytacji, mebli oraz opłat wygląda analogicznie, z zastrzeżeniem dodawania dla właściciela (lokator może dodać zapłaty).

- Powiadomienia widoczne ze strony lokatora, na których (Rys. 7) widać nowe umowy zaproponowane przez właścicieli, wizytacje i awarie w mieszkaniach z aktywnymi umowami. Umowy w tej perspektywie wyglądają trochę inaczej, jako że mieszkania prowadzą tylko do jednej umowy.



Rysunek 5: Przeglądanie i tworzenie mieszkań



Rysunek 6: Przeglądanie umów oraz komunikator

3.4 Testy

Aplikacja została przetestowana manualnie za pomocą 6 różnych urządzeń, zarówno realnych, jak i wirtualnych włączanych za pomocą Android Studio. Dodatkowo przy pomocy ochotników (i ich urządzeń) udało się usprawnić pewne funkcjonalności, oraz poprawić wygląd. Serwer jak już zostało wspomniane był testowany przy pomocy programu Postman, aby pominąć możliwe błędy w interpretacji na telefonie. Baza była sprawdzana manualnie pod kontem poprawnych danych, korzystając z MySQL workbench, który pozwala na wykorzystanie graficznego środowiska do łatwiejszego sprawdzania widoków oraz tabel.

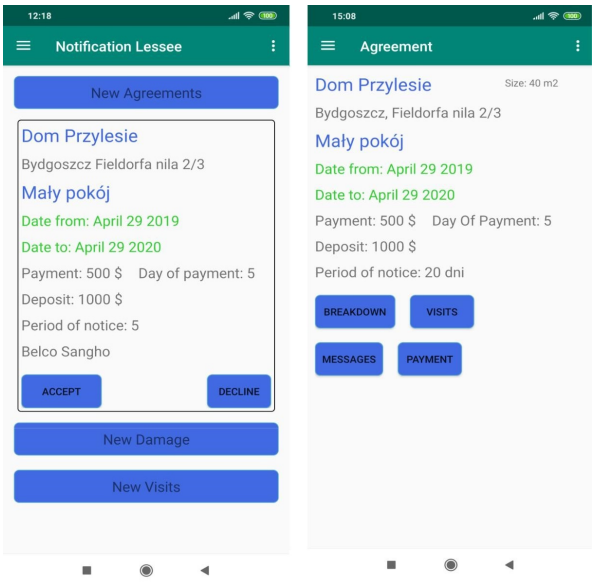
4 Wnioski

Aplikacja mobilna pozwala efektywnie zarządzać mieszkaniami poprzez zawarte w niej wiele opcji. Jednakże aby móc wprowadzać na rynek ten program, należałoby się uważniej przyjrzeć systemowi szyfrowania i autoryzacji, który musiałby spełniać wysokie standardy, z uwagi na przechowywanie wielu newralgicznych danych, takich jak adresy mieszkań, czy kwoty opłat ponoszonych z tytułu najmu, bądź imiona i nazwiska osób powiązanych umową. W przypadku dalszego rozwoju produktu, będzie to najważniejszy następny etap badań.

Jako że finalny produkt został stworzony w środowiskach zapewniających szeroką gamę rozszerzeń oraz bibliotek tworzonych przez środowisko programistyczne (Android Studio oraz Node.js), istnieje duży potencjał na rozwój poza aspektem bezpieczeństwa. Kilka kolejnych potencjalnych pól rozwoju to:

- Możliwość dalszego podziału mieszkań na pokoje.
- Podpisywanie umów przez wiele osób, oraz podział opłat zgodnie z ustaloną kwotą.
- Wyświetlanie powiadomień nawet kiedy aplikacja jest wyłączona, aby zmniejszyć potrzebę niepotrzebnego wchodzenia do aplikacji.
- Możliwość wykonywania płatności internetowych poprzez program.

Jak widać więc istnieje wiele możliwych usprawnień, które mogłyby podnieść jakość aplikacji.



Rysunek 7: Przeglądanie powiadomień oraz umowy lokatora

Literatura

[1] Nathalie Girouard, Mike Kennedy, Paul van den Noord, and Christophe André. Recent house price developments. (475), 2006.

[2] Krzysztof Sacha. *Inżynieria Oprogramowania*. PWN Warszawa, 2010.

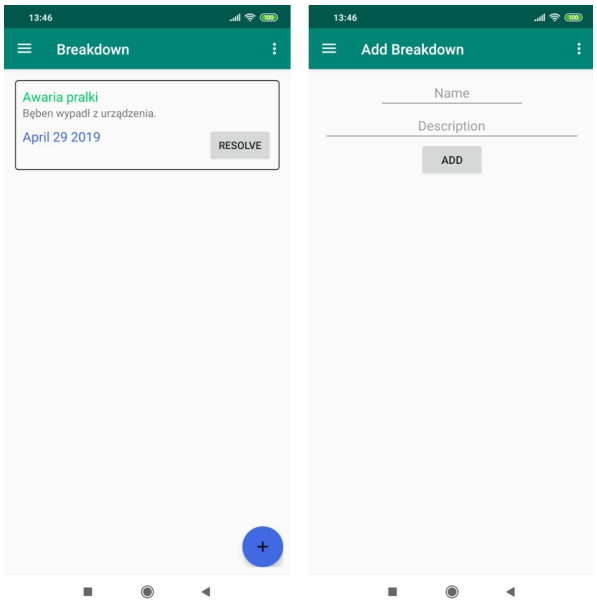
[3] Oracle. *MySQL Workbench Manual*, 65006 edition, 02 2020.

[4] *Node.js Manual Guides*.

[5] StrongLoop IBN. *Express Routing Manual*.

[6] Google. *Android Views*.

[7] Heroku. *Getting Started on Heroku with Node.js*.



Rysunek 8: Przeglądanie i dodawanie awarii