



# Politechnika Wrocławska

Wydział Informatyki i Zarządzania

Kierunek studiów: Informatyka

Praca dyplomowa – inżynierska

## APLIKACJA MOBILNA WSPOMAGAJĄCA KOMUNIKACJĘ RZEMIEŚLNIKA Z UŻYTKOWNIKIEM JEGO WYROBÓW

Dominik Ilski

słowa kluczowe:  
Flutter, Android, aplikacja mobilna, Firebase, rzemieślnik, komunikacja, lokalizacja

### krótkie streszczenie:

W pracy przedstawiono projekt oraz implementację aplikacji, służącej do wspomagania komunikacji rzemieślnika z użytkownikiem jego wyrobów. W aplikacji zastosowano między innymi dodawanie ofert na przedmioty i usługi, komunikator tekstowy, dostosowanie własnego warsztatu rzemieślniczego oraz możliwość sprawdzenia statusu zlecenia.

Opiekun pracy dyplomowej	dr inż. Zbigniew Staszak		
	Tytuł/stopień naukowy/imię i nazwisko		
Ostateczna ocena za pracę dyplomową			
Przewodniczący Komisji egzaminu dyplomowego			
	Tytuł/stopień naukowy/imię i nazwisko	ocena	podpis

*Do celów archiwalnych pracę dyplomową zakwalifikowano do:\**

- a) kategorii A (akta wieczyste)
- b) kategorii BE 50 (po 50 latach podlegające ekspertyzie)

*\* niepotrzebne skreślić*

pieczętka wydziałowa

Wrocław 2023

# Spis treści

<b>Streszczenie</b> . . . . .	3
<b>1. Wstęp</b> . . . . .	4
1.1. Wprowadzenie do problematyki . . . . .	4
1.2. Geneza pracy . . . . .	4
1.3. Cel pracy . . . . .	5
1.4. Zakres pracy . . . . .	5
<b>2. Przegląd podobnych istniejących rozwiązań</b> . . . . .	6
2.1. Reprezentatywne rozwiązania . . . . .	6
2.1.1. Mapy . . . . .	6
2.1.2. Fixly - do usług! . . . . .	8
2.1.3. Fixly dla wykonawców - zdobywaj zlecenia . . . . .	10
2.1.4. OLX - ogłoszenia lokalne . . . . .	11
2.1.5. Warszawscy Rzemieślnicy . . . . .	12
2.2. Zestawienie zalet i wad . . . . .	13
2.2.1. Mapy . . . . .	13
2.2.2. Fixly - Do usług! . . . . .	14
2.2.3. Fixly dla wykonawców - zdobywaj zlecenia . . . . .	14
2.2.4. OLX - ogłoszenia lokalne . . . . .	15
2.2.5. Warszawscy Rzemieślnicy . . . . .	15
2.3. Podsumowanie . . . . .	16
<b>3. Przegląd dostępnych technik oraz platform programistycznych</b> . . . . .	17
3.1. Warstwa prezentacji . . . . .	17
3.1.1. React Native . . . . .	17
3.1.2. Flutter . . . . .	18
3.1.3. Ionic . . . . .	19
3.1.4. Podsumowanie . . . . .	19
3.2. Warstwa logiki . . . . .	19
3.2.1. Amazon AWS . . . . .	20
3.2.2. Google Firebase . . . . .	21
3.2.3. Flotiq . . . . .	22
3.2.4. Ghost . . . . .	23
3.2.5. Strapi . . . . .	24
3.3. Użyte technologie . . . . .	24

3.4. Narzędzia pomocnicze . . . . .	25
3.4.1. System kontroli wersji . . . . .	25
3.4.2. IDE . . . . .	26
3.4.3. System operacyjny komputera . . . . .	26
3.4.4. Emulator urządzenia . . . . .	27
3.4.5. Narzędzie do prototypowania . . . . .	27
3.5. Podsumowanie . . . . .	27
<b>4. Projekt . . . . .</b>	<b>28</b>
4.1. Wymagania projektowe . . . . .	28
4.2. Wymagania funkcjonalne . . . . .	29
4.3. Wymagania niefunkcjonalne . . . . .	30
4.4. Model przypadków użycia . . . . .	30
Spis Przypadków Użycia . . . . .	31
4.4.1. Diagram przypadków użycia . . . . .	31
4.4.2. Scenariusze przypadków użycia . . . . .	33
4.5. Model domenowy . . . . .	46
4.6. Model fizyczny bazy danych . . . . .	47
4.7. Prototyp interfejsu użytkownika . . . . .	50
4.8. Diagram rozmieszczenia . . . . .	53
4.9. Wzorce projektowe . . . . .	53
4.10. Podsumowanie . . . . .	53
<b>5. Implementacja . . . . .</b>	<b>54</b>
5.1. Warstwa Prezentacji . . . . .	54
5.1.1. Najważniejsze rozwiązania . . . . .	54
5.1.2. Napotkane problemy oraz sposób ich rozwiązania . . . . .	54
5.2. Warstwa Logiki . . . . .	54
5.2.1. Najważniejsze rozwiązania . . . . .	54
5.2.2. Napotkane problemy oraz sposób ich rozwiązania . . . . .	54
5.3. Testy . . . . .	54
5.3.1. Scenariusze testowe . . . . .	54
5.3.2. Wyniki testów . . . . .	54
5.4. Prezentacja Aplikacji . . . . .	54
5.5. Podsumowanie . . . . .	54
<b>6. Zakończenie . . . . .</b>	<b>55</b>
6.1. Zrealizowane prace . . . . .	55
6.2. Dalsze kroki rozwoju . . . . .	55
<b>Bibliografia . . . . .</b>	<b>56</b>
<b>Spis rysunków . . . . .</b>	<b>60</b>
<b>Spis tabel . . . . .</b>	<b>61</b>

## **Streszczenie**

Celem pracy było wytworzenie aplikacji służącej do komunikacji rzemieślnika z odbiorcą jego wyrobów. Na rynku brakowało rozwiązania łączącego wszystkie funkcje, które są niezbędne w komunikacji oraz wszystkich rzeczy z nią powiązanych. Są to między innymi: chat pomiędzy użytkownikiem, a mistrzem, lokalizacja zakładów rzemieślniczych, personalizacja swojego warsztatu.

W ramach pracy przygotowano aplikację, spełniającą te funkcje, wykorzystującą framework Flutter. Dane użytkowników oraz rzemieślników przechowywane są w bazie danych Cloud Firestore oraz serwisie typu headless CMS, Flotiq. Wszystkie funkcje systemu zostały zawarte w mikrousługach, a udostępnione są one przez interfejs REST API.

Oprócz projektu aplikacji praca zawiera wyniki testów użyteczności przeprowadzonych przez krewnych oraz znajomych.

Praca przygotowana w ramach projektu inżynierskiego może zostać wykorzystana przez rzemieślników, chcących wyjść w stronę klienta oraz użytkowników chcących skorzystać z usług mistrzów.

## **Abstract**

The aim of the work was to create an application for communication between the craftsman and the recipient of his products. There was no solution on the market that would combine all the functions that are necessary in communication and all the things related to it. These are among others: chat between the user and the master, the location of craft plants, personalization of your workshop.

As part of the work, an application was prepared that fulfills these functions, using the Flutter framework. Data of users and craftsmen are stored in the Cloud Firestore database and the headless CMS, Flotiq website. All system functions are contained in microservices and made available through the REST API.

In addition to the application design, the work contains the results of usability tests carried out by relatives and friends.

Work prepared as part of an engineering project can be used by craftsmen who want to go out to the client and users who want to use the services of the masters.

## **1. Wstęp**

### **1.1. Wprowadzenie do problematyki**

Przed erą druku, komunikacja między ludźmi była głównie słowna. Niewiele osób potrafiło czytać i pisać. Żeby zdobyć jakieś produkty, trzeba było udać się do miasta. Ludziom zależało, by skomunikować się z wytwórcami, gdyż nie mieli innego sposobu na uzyskanie jakiegoś produktu. Wraz z biegiem czasu, pojawieniem się mediów elektronicznych, zapoczątkowaniem masowej produkcji, osoby które potrzebują jakiś przedmiot nie muszą już ruszać się z domu. Mogą zamówić produkt przez Internet, używając na przykład aplikacji mobilnej. Aktualnie dostępny jest duży wybór mediów elektronicznych zapewniających dostęp do przedmiotów i do usług. Jednak jedna z branży nie rozwinęła się wystarczająco w tej dziedzinie – są to rzemieślnicy. Według wiedzy autora tej pracy, do dzisiaj nie powstało żadne z dedykowanych rozwiązań dla mistrzów, którzy z biegiem czasu przestali być jednoosobowym producentem różnego rodzaju przedmiotów. Brakuje medium, które wspomagałby komunikację rzemieślnika z odbiorcą jego produktów, a dodatkowo przenosiłoby ją na urządzenie, które posiada prawie każdy, czyli telefon komórkowy. Niniejsza praca odniesie się do opisanej powyżej problematyki. W pracy będą rozróżniane dwa pojęcia. Pierwszym z nich jest użytkownik, a drugim rzemieślnik. Użytkownik to biorca usług, przedmiotów wykonywanych przez rzemieślników, nazywany będzie również usługobiorcą oraz odbiorcą. Drugim pojęciem jest rzemieślnik - osoba, świadcząca usługi, wytwarzająca przedmioty. Będzie on w tej pracy również nazywany mistrzem oraz usługodawcą.

### **1.2. Geneza pracy**

Początkowo planowanym tematem pracy była „Aplikacja wspomagająca budowę społeczności fanów ręczne robionych noży”. Temat ten był związany z zapotrzebowania autora na aplikację, która w przyszłości pomogłaby promować oraz rozwijać hobby autora związane z produkcją noży japońskich. Po konsultacjach z promotorem pracy uznano, że temat jest zbyt niszowy, stąd też pomysłem rozwijającym ową ideę było rozszerzenie wspomnianej myśli do wszystkich rzemieślników, a nie ograniczenie się do tylko jednego cechu. W ten sposób został sformułowany aktualny temat pracy. Ludzie coraz rzadziej korzystają z usług rzemieślników. Często jest to spowodowane faktem, że większość informacji przepływa dziś przez media elektroniczne, do których większość mistrzów rzadko zagląda. Korzystanie z ich usług, jest nie tylko ekologiczne, ale jest również wsparciem dla zanikających zawodów. Analiza rynku

opisana w drugim rozdziale niniejszej pracy, pokazuje, że aktualnie na rynku brakuje rozwiązania, które odzwierciedlałyby założenia projektu.

### **1.3. Cel pracy**

Celem pracy jest zaprojektowanie i wykonanie aplikacji mobilnej wspomagającej komunikację rzemieślnika z użytkownikiem jego wyrobów.

Powyższy cel nie jest czysto informacyjny, ani czysto komercyjny. Jest także promocją kultury miejskich rzemieślników, których z roku na rok jest coraz mniej.

### **1.4. Zakres pracy**

Zakres pracy jest określony przez kolejne jej rozdziały. I tak po niniejszym rozdziale zawierającym informacje wstępne, w drugim rozdziale zostaną przedstawione rozwiązania konkurencyjne wobec projektowanej w ramach tej pracy aplikacji wraz z ich krytyczną analizą, z której wnioski będą podstawą decyzji o potrzebie realizacji tej aplikacji. Trzeci rozdział będzie dotyczył przeglądu technologii. Dokładna i krytyczna analiza dostępnych framework'ów, szkieletów budowy aplikacji, zarówno z warstwy prezentacji jak i z warstwy logiki, pozwoli na wybranie narzędzi najbardziej pasujących do założeń projektowych. Czwarty rozdział będzie przedstawał projekt aplikacji. W nim zawarta zostanie najważniejsza część tej pracy. Przedstawić będzie wymagania projektowe, funkcjonalne, niefunkcjonalne, model przypadków użycia, model domenowy, diagram rozmieszczenia oraz prototypy interfejsu użytkownika. Przedostatni, piąty rozdział poświęcony zostanie implementacji przedstawionego wcześniej projektu, przy użyciu opisanych w rozdziale trzecim narzędzi programistycznych. Przedstawione w nim zostaną najważniejsze, z punktu widzenia autora, rozwiązania oraz napotkane podczas implementacji problemy. Całość zwieńczona zostanie zakończeniem, przedstawiającym zrealizowane prace oraz dalsze kroki rozwoju.

## **2. Przegląd podobnych istniejących rozwiązań**

W tym rozdziale pracy, szczególna uwaga zostanie zwrócona na aktualny stan rynku aplikacji mobilnych oraz webowych związanych z dziedziną pracy. Aplikacje te, po pierwsze nie tylko mogą stanowić konkurencję dla nowo tworzonego projektu, ale również mogą być inspiracją do rozwijania rozwiązań, które zostały w ich ramach zawarte. Zapoznanie się z konkurencją, umożliwia również użycie oraz ulepszenie sprawdzonych na rynku rozwiązań z zakresu *UI (user interface)*, czyli interfejsu użytkownika oraz *UX (user experience)*, czyli doświadczenia użytkownika związane z aplikacją. Użytkownicy, a ogólniej ludzie, lubią wszystko co jest im już znane. Przedstawienie im tego w nowej, odświeżonej formie, może stanowić mały, ale równie ważny krok w stronę sukcesu projektowanej aplikacji mobilnej.

### **2.1. Reprezentatywne rozwiązania**

Po przeglądzie rynku ustalone kilka bardzo ważnych faktów. Po pierwsze, brakuje rozwiązania, które w pełni odpowiadały celom projektu. Oznacza to, że aktualnie na rynku brakuje rozwiązań, które łączyłyby potrzeby rzemieślnika, jak i odbiorcy jego wyrobów, w jednej aplikacji. Znalezienie niszy znacznie zwiększa prawdopodobieństwo popularności tworzonej aplikacji mobilnej. Aktualnie na rynku dostępne jest kilka rozwiązań, które w pewnym stopniu pokrywają się z celem pracy i które autor tego dokumentu uznał za reprezentatywne, mogą więc stanowić one inspirację podczas projektowania. Rozwiązania te, mogą zostać użyte jako zastępce do nowo tworzonej aplikacji mobilnej, jednak nie proponują one tak skondensowanego i pełnego rozwiązania, a co za tym idzie nie tworzą całości, która może zostać doceniona przez użytkownika. Wspomniane aplikacje to:

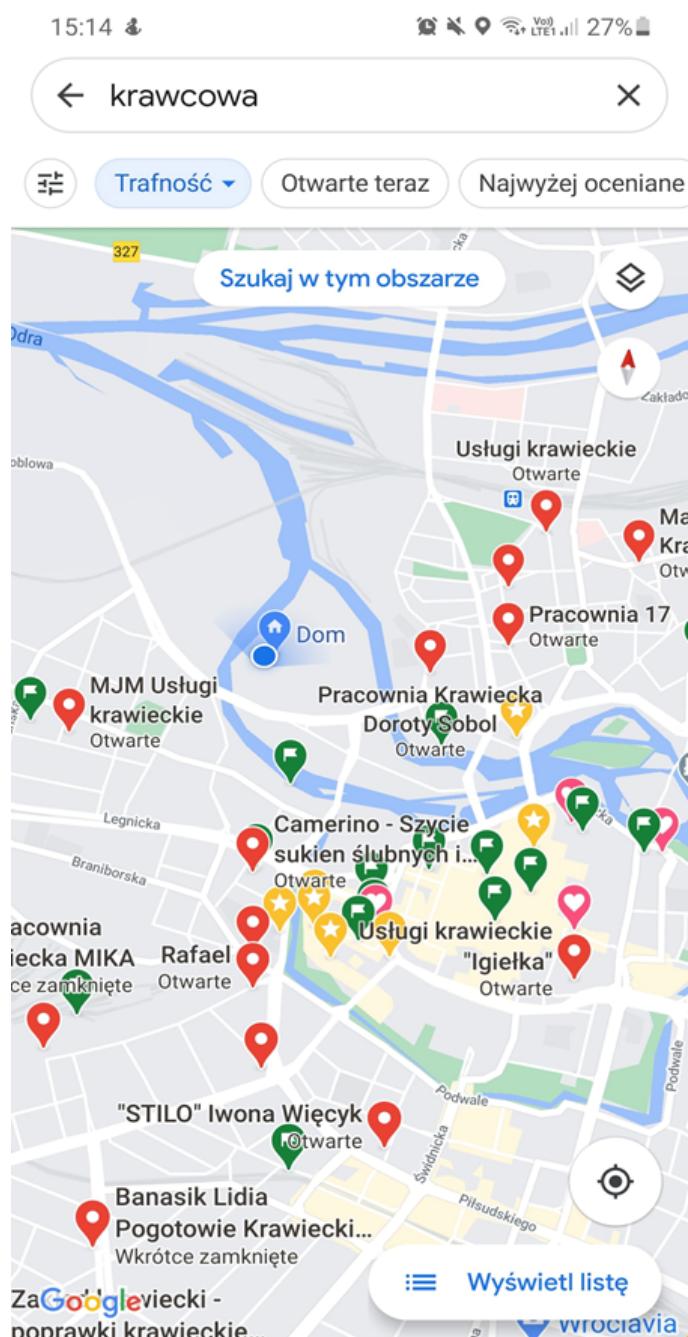
- Mapy [45],
- Fixly – do usług! [57],
- Fixly dla wykonawców - zdobywaj zlecenia [56],
- OLX - ogłoszenia lokalne [58],
- Warszawscy Rzemieślnicy [52].

W poniższych podrozdziałach zostaną przedstawione ich funkcje wraz krytyczną ich analizą.

#### **2.1.1. Mapy**

Mapy [45] to aplikacja służącą do wszystkiego co związane jest z lokalizacją i nawigacją. Jest szeroko używana na świecie z wynikiem 5 mld pobrań w samym Sklepie

Play. Znana jest prawie każdemu użytkownikowi telefonu z Androidem. Rozwiążanie pozwala na wyszukiwanie, dodawanie oraz zapisywanie różnego rodzaju miejsc. Mapy ma szereg zastosowań i jest zdecydowanym liderem na rynku w dziedzinie nawigacji oraz lokalizacji. Aplikacja sama w sobie nie posiada szczególnych wad. Jednak, gdyby używać jej do celów, takich jak temat tej pracy, okazuje się, że dostarcza ona użytkownikowi bardzo dużo szumu informacyjnego. Dobra to widać na przykładzie (rys. 2.1). Użytkownik, próbuje znaleźć krawcową. W aplikacji Mapy wpisuje frazę *krawcowa*, otrzymuje poniższy wynik (rys. 2.1).

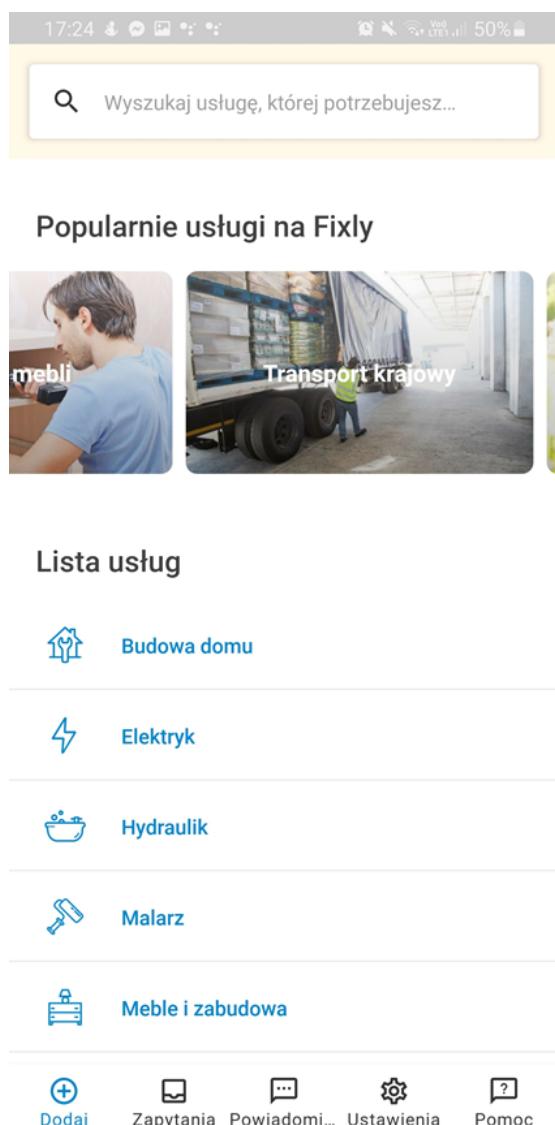


Rys. 2.1. Wynik wyszukiwania w aplikacji Mapy [45]

Aplikacja pokazuje oczekiwane miejsca, jednak oprócz nich pokazuje jeszcze dużo innych, niezwiązanych z wyszukiwaniem, symboli. Często są to również reklamy. Zmniejsza to czytelność otrzymanych wyników, co jest dużą wadą. Aby aplikacja mogła pełnić rolę, jako mapa pokazująca tylko rzemieślników w okolicy , należy ją wyspecjalizować. Ze względu na to, że Mapy, to otwarty standard, w projekcie zostanie użyte *SDK* [43]. Zestaw narzędzi programistycznych, który umożliwia tworzenie aplikacji dla określonego pakietu oprogramowania [60].

### 2.1.2. Fixly - do usług!

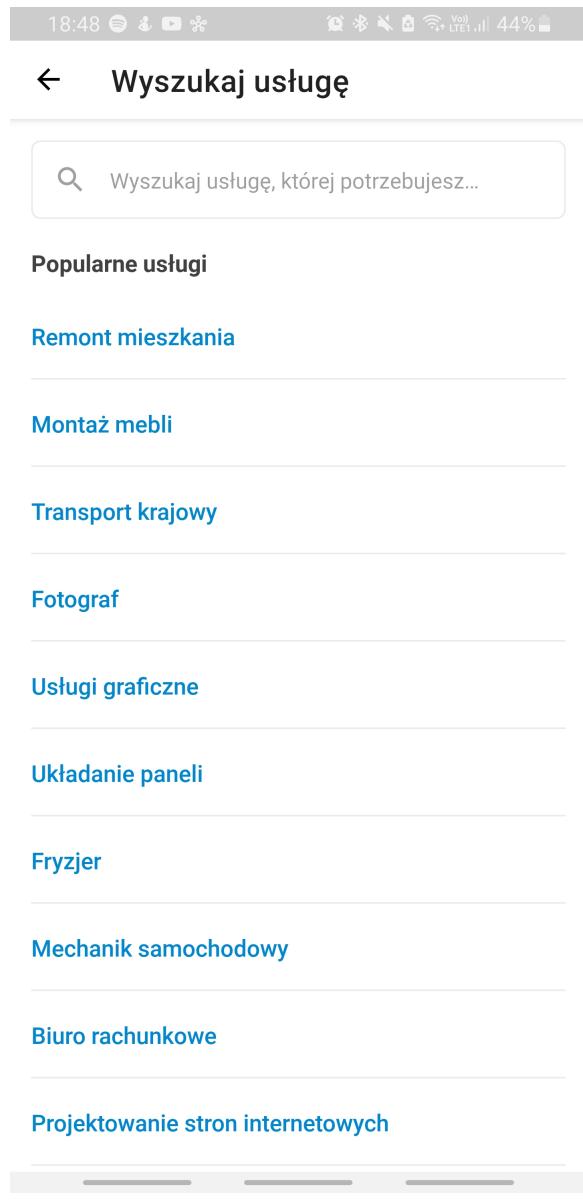
Fixly – do usług! [57] jest względnie nową aplikacją, która pozwala na znalezienie wykonawców różnych rodzajów usług. Jak pokazuje rys. 2.2.



Rys. 2.2. Przegląd dostępnych kategorii w aplikacji Fixly [57]

Wyszukiwanie zleceniobiorców rozpoczyna się do wybrania konkretnej kategorii. Następnie doprecyzowane są szczegółowe zlecenia. Po uzupełnieniu formularza, wystawiane jest ogłoszenie, do którego mogą zgłosić się zleceniobiorcy. Po interakcji z aplikacją udało ustalić się pewne fakty:

- stworzenie zgłoszenia, na przykład na malowanie ścian, wymaga 12 pojedynczych interakcji z aplikacją, tym samym przechodząc w tym czasie przez 4 różne ekran,
- naciśnięcie na głównym ekranie aplikacji (rys. 2.2) paska wyszukiwania, przenosi do osobnego ekranu (rys. 2.3), mimo tego, że komponent sugeruje możliwość wpisania tekstu.

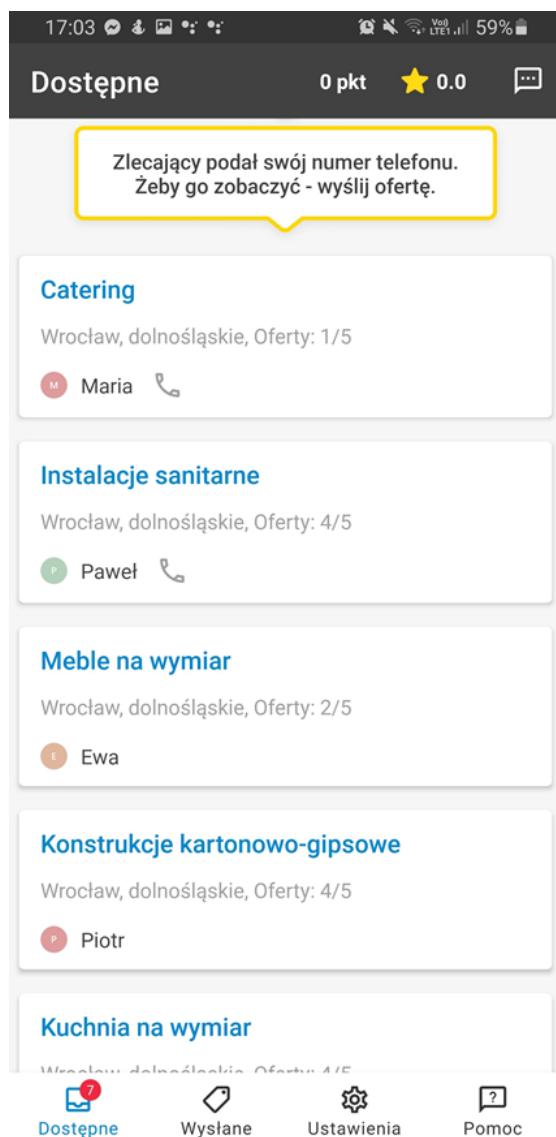


Rys. 2.3. Ekran wyszukiwania w aplikacji Fixly [57]

Rozwiązań Fixly dzieli się na dwie osobne aplikacje. Jedna z nich to aplikacja odpowiedzialna, za stronę klienta usług, a druga jest odpowiedzialna za stronę wykonawcy owych usług. Podział ten jest bardzo dobrze przemyślany, gdyż osoba, która wyświadczenie usługi może też być odbiorcą usług innych użytkowników. Dodatkowo z punktu widzenia konserwacji oprogramowania, ułatwia to pracę deweloperów. Aplikacja uniemożliwia jednak założenie konta na jeden e-mail na różnych platformach. Wymaga to od użytkownika założenia nowego konta e-mail, gdy ten chce zostać usługodawcą. Część dla usługodawców zostanie opisana w następnym punkcie. Dużą zaletą aplikacji jest brak reklam.

### 2.1.3. Fixly dla wykonawców - zdobywaj zlecenia

Aplikacja Fixly dla usługodawców [56] jest minimalistyczna. Zawiera w sobie tylko najpotrzebniejsze funkcje rys. 2.4.

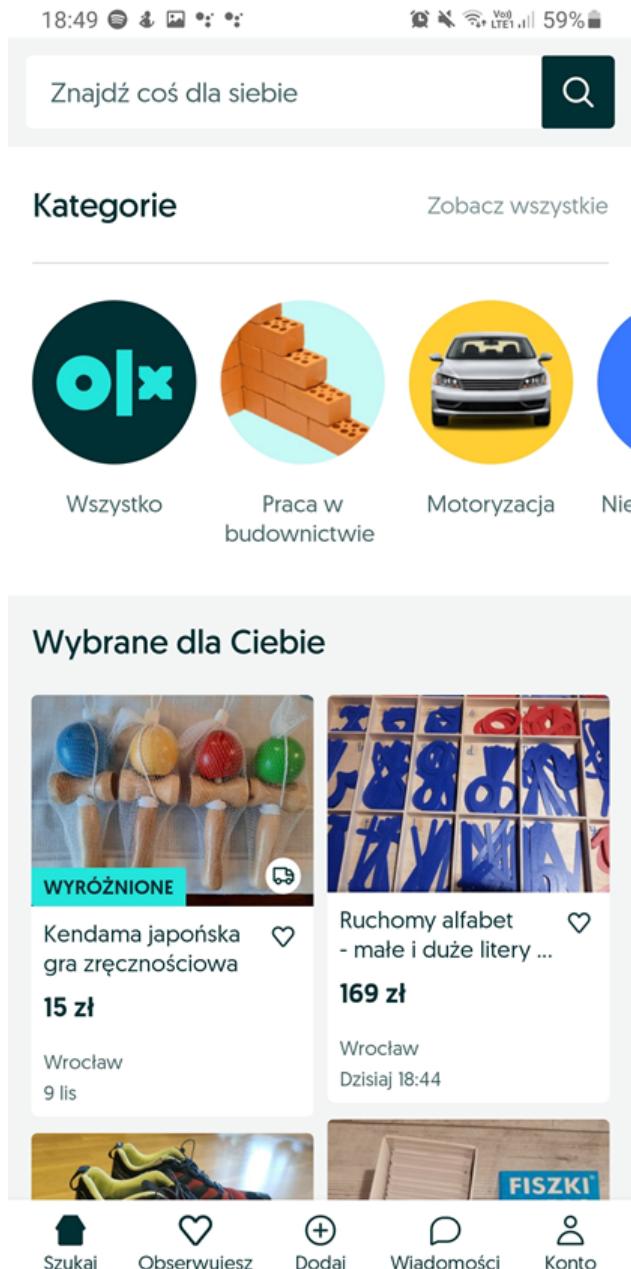


Rys. 2.4. Wygląd aplikacji Fixly dla wykonawców [56]

Użytkownik aplikacji, musi się zarejestrować, a podczas niej musi podać rodzaje zleceń, których będzie chciał się podejmować. Po pełnej konfiguracji, użytkownikowi wyświetlać się będą dostępne zlecenia rys. 2.4, których szczegóły może ustalić już ze zleceniodawcą. Główną funkcją aplikacji jest odbieranie nowych zleceń. Minimalistyczne *UI* aplikacji wzbogacone jest jednak przez system powiadomień. Aplikacja posiada również system oceny jakości pracy zleceniodawców. Są one wystawiane przez zleceniodawców. Ta cecha jest bardzo ważnym elementem opisywanego rozwiązania konkurencyjnego. Dzięki niemu jakość będzie rosła, gdyż klienci częściej będą korzystać z lepiej ocenianych specjalistów, a tych ocenianych gorzej będą omijać. Wadą rozwiązania jest fakt, że do konkretnego ogłoszenia może zgłosić się tylko 5 wykonawców, powoduje to przypadki, w których wykonawca musi ciągle monitorować aplikację, by odpowiednio szybko zgłosić się do ogłoszenia. Mikropłatności w aplikacji, zwiększą szansę na zdobycie zlecenia. Może to powodować przypadki, w których lepiej oceniani, przez społeczność wykonawcy nie będą proponowani, jako pierwsi. Duża zaletą aplikacji jest też brak reklam.

#### **2.1.4. OLX - ogłoszenia lokalne**

OLX - ogłoszenia lokalne [58] znane wcześniej jako „tablica.pl”. jest bardzo popularną aplikacją do wstawiania oraz wyszukiwania ogłoszeń. OLX (rys. 2.5), jest ważnym odniesieniem dla całego projektu, gdyż ogłoszenia nie ograniczają się tylko do przedmiotów. To co tam zostanie umieszczone określa autor. Aplikacja prócz wspomnianej możliwości, pozwala również na dodawanie ogłoszeń do obserwowanych. Możliwy jest również chat z innymi użytkownikami, ocena ogłoszeniodawców oraz konfigurację konta użytkownika. Te funkcje sprawiają, że jest to aplikacja najbardziej zbliżona do założeń tworzonego projektu. Zastosowanie rozwiązań użytych w OLX, może pozytywnie wpływać na późniejszą popularność przedstawianego rozwiązania. Ilość dostępnych na OLX ogłoszeń [58], może powodować trudności, w rozróżnieniu, mistrza od zwykłego użytkownika. Dodatkowo w OLX brakuje mapy, pozwalającej zlokalizować sprzedających.

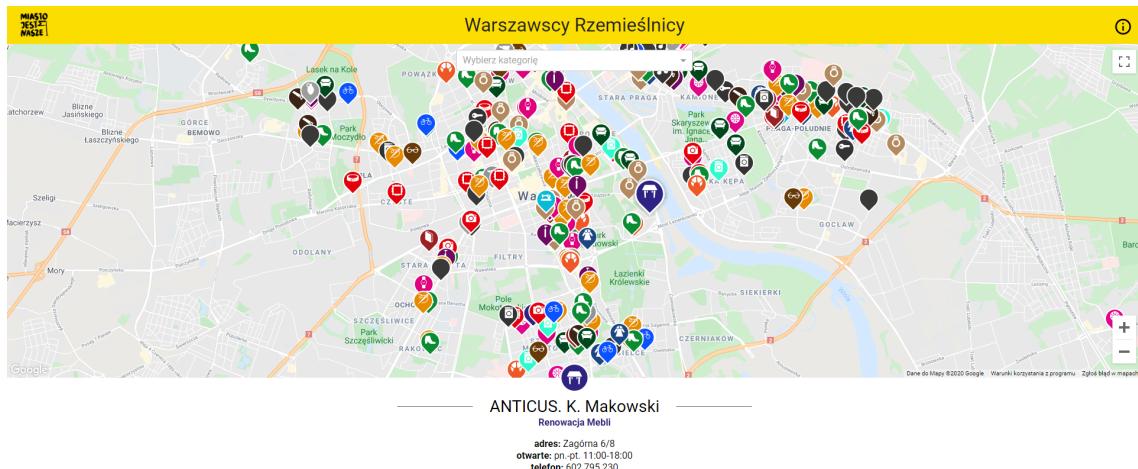


Rys. 2.5. Wygląd aplikacji OLX [58]

### 2.1.5. Warszawscy Rzemieślnicy

Warszawscy Rzemieślnicy [52] jest stroną internetową, na której znajduje się mapa zawierająca zebrane lokalizacje rzemieślników, działających w Warszawie. Rozwiążanie to jest różne od dotychczas przedstawionych, gdyż jako jedyne nie było projektem komercyjnym, a społecznym. Projekt ten jest prowadzony przez Stowarzyszenie Miasto Jest Nasze od 2015 roku [52]. Ma on na celu kultywowanie rzemiosła i wspieranie rzemieślników, jako elementu tożsamości Warszawy. Storna jest minimalistyczna (rys. 2.6), zawiera tylko najpotrzebniejsze informacje, nie daje

ona możliwości, tworzenia konta użytkownika, komunikacji z rzemieślnikiem. Na stronie zamieszczone są tylko najpotrzebniejsze informacje.



Rys. 2.6. Wygląd strony internetowej Warszawscy Rzemieślnicy [52]

## 2.2. Zestawienie zalet i wad

Przedstawione w powyższych podrozdziałach aplikacje, różnią się od siebie w znaczącym stopniu, stąd trudno je w sposób jednolity porównać. W tym podrozdziale przedstawione zostaną, w tabelach 2.1 - 2.5, najważniejsze wady oraz zalety każdej z wybranych aplikacji a także sprecyzowane zostaną wnioski z tego zestawienia wynikające.

### 2.2.1. Mapy

Mapy, nawet jako częściowe rozwiązanie, nie dostarczają wystarczająco skondensowanej treści, by uznać je za realną konkurencję projektu będącego tematem tej pracy. Rozważenie tego rozwiązania, jako konkurencyjnego do projektowanego w tej pracy, miało głównie na celu sprawdzenie jego jakości oraz poznanie możliwości użycia dostępnego SDK w nowo tworzonej aplikacji. Wnioski z analizy, jak i samo SDK zostanie zastosowane w tym projekcie.

Tabela 2.1. Wady i zalety aplikacji Mapy

Mapy	
Zalety	Wady
<ul style="list-style-type: none"> <li>- Duża część rynku.</li> <li>- Szeroko używana również w innych aplikacjach.</li> <li>- Znana przez użytkowników.</li> </ul>	<ul style="list-style-type: none"> <li>- Szum informacyjny.</li> <li>- Reklamy, związane z promowanymi lokalizacjami</li> </ul>

### **2.2.2. Fixly - Do usług!**

Aplikacja Fixly, opiera się głównie na usługach, nie stanowi więc ona konkurencji dla nowo tworzonego projektu. Główną cechą aplikacji jest sposób zgłaszania potrzeby na wykonanie usługi. Przez to pojawia się przypadek, w którym nie ma możliwości kontaktu z wykonawcą. W aplikacji znajduje się zakładka z pomocą, jest to ogólnie dobra cecha, jednak może świadczyć o tym, że aplikacja jest nieintuicyjna. Kolejną wadą jest fakt, że użytkownik może otrzymać tylko 5 ofert od wykonawców. Aplikacja nie jest też zbyt popularna, gdyż ma tylko 50 tys. pobrań w Sklepie Play [57].

Tabela 2.2. Wady i zalety aplikacji Fixly - Do usług!

Fixly - Do usług!	
Zalety	Wady
<ul style="list-style-type: none"><li>- Znacznie więcej usługodawców niż usługobiorców.</li><li>- Możliwość wybory specjalisty, z tych co się zgłosili.</li><li>- Brak reklam.</li><li>- Zakładka z instrukcjami, dla użytkownika.</li></ul>	<ul style="list-style-type: none"><li>- Brak możliwości bezpośredniego kontaktu ze specjalistą, bez tworzenia ogłoszenia.</li><li>- Mało funkcji.</li><li>- Możliwość otrzymania tylko 5 ofert.</li></ul>

### **2.2.3. Fixly dla wykonawców - zdobywaj zlecenia**

Aplikacja Fixly dla wykonawców jest dwa razy bardziej popularna, niż aplikacja dla użytkowników, posiada około 100 tysięcy pobrań [56]. Oznacza to, że problem analizowanego rozwiązania istnieje po stronie aplikacji dla usługobiorców. Stąd też przy pracy nad niniejszym projektem, szczególną uwagę warto zwrócić właśnie na stronę użytkownika, gdyż mimo wspomnianych wad, Fixly dla wykonawców jest bardziej popularna mimo tego, że wykonawców na rynku jest mniej, niż potencjalnych odbiorców usług.

Tabela 2.3. Wady i zalety aplikacji Fixly dla wykonawców - zdobywaj zlecenia

Fixly - Do usług!	
Zalety	Wady
<ul style="list-style-type: none"><li>- Prosta w obsłudze aplikacja.</li><li>- Powiadomienia push.</li><li>- Oceny użytkowników.</li><li>- Brak reklam.</li><li>- Możliwość zdobywania certyfikatów.</li></ul>	<ul style="list-style-type: none"><li>- Brak możliwości założenia konta na ten sam e-mail co konto użytkownika</li><li>- Mało funkcji.</li><li>- Do ofert trzeba zgłaszać się samemu, kto pierwszy ten lepszy.</li><li>- Mikropłatności w aplikacji.</li></ul>

#### **2.2.4. OLX - ogłoszenia lokalne**

Opisywana aplikacja odniósła ogromny sukces, gdyż została pobrana 10 mln razy [58]. Jednak, mimo podobieństw jakie można znaleźć pomiędzy projektem, a analizowanym rozwiązaniem, brakuje w niej kilku ważnych elementów (tab. 2.4). Aplikacja będzie ważnym punktem odniesienia dla realizowanego projektu, z powodu jej popularność, jednak nie stanowi ona bezpośredniej konkurencji, gdyż jest bardziej nastawiona na same ogłoszenia, a nie na usługodawców, którymi w niniejszej pracy są rzemieślnicy.

Tabela 2.4. Wady i zalety aplikacji OLX

Fixly - Do usług!	
Zalety	Wady
<ul style="list-style-type: none"><li>- Dużo funkcji.</li><li>- Dowolność w tworzeniu ogłoszeń.</li><li>- Dodawanie ogłoszeń do obserwowanych.</li><li>- Brak reklam.</li><li>- Chat z ogłoszeniodawcą.</li><li>- Duża opcja konfiguracji wyszukiwania.</li><li>- Oceny sprzedających.</li></ul>	<ul style="list-style-type: none"><li>- Brak możliwości sprawdzenia okolicznych ogłoszeń na mapie.</li><li>- Brak możliwości personalizacji własnego sklepu.</li><li>- Brak możliwości wyszukiwania po użytkowniku.</li></ul>

#### **2.2.5. Warszawscy Rzemieślnicy**

Przedstawiona strona internetowa nie jest komercyjnym projektem, stąd też jest mało rozbudowana. Jednak przedstawione w niej funkcje bardzo dobrze odzwierciedlają te, które zawarte zostaną w projekcie. Strona będzie ważnym punktem odniesienia podczas pracy nad aplikacją.

Tabela 2.5. Wady i zalety aplikacji Warszawscy Rzemieślnicy

Fixly - Do usług!	
Zalety	Wady
<ul style="list-style-type: none"><li>- Możliwość wyboru specjalności rzemieślnika.</li><li>- Ikony na mapie określające typ rzemieślnika.</li><li>- Na mapie pokazywani są tylko rzemieślnicy.</li></ul>	<ul style="list-style-type: none"><li>- Brak możliwości dodania warsztatu ręcznego.</li><li>- Brak możliwości wyszukiwania.</li><li>- Mapa ogranicza się tylko do Warszawy.</li><li>- Brak możliwości lokalizacji na mapie.</li></ul>

### **2.3. Podsumowanie**

Przegląd dostępnych na rynku rozwiązać pozwolił wyznaczyć kilka ważnych funkcji, które projektowana aplikacja powinna posiadać:

- tworzenie ofert,
- personalizacja sklepu,
- system ocen,
- chat,
- mapa z zaznaczonymi warsztatami,
- wyszukiwanie po ofertach oraz rzemieślnikach,
- lokalizacja pobliskich warsztatów rzemieślniczych.

Każda z przedstawionych aplikacji, posiadała któryś z tych cech. Jednak, żadna nie łączyła ich wszystkich. Najbliżej przedstawianego projektu znajduje się OLX, jest on również najbardziej popularny. Stąd też będzie dobrą bazą przy projektowaniu *UI* oraz *UX* aplikacji.

### **3. Przegląd dostępnych technik oraz platform programistycznych**

W tym rozdziale poruszona zostanie kwestia technologii oraz rozwiązań, jakie zostały użyte do realizacji aplikacji. Rozpocznie się on od przedstawienia technologii nadających się potencjalnie do zastosowania w projekcie, a następnie zostaną wyciągnięte wnioski oraz przedstawione zostanie uzasadnienie wyboru konkretnych rozwiązań. Modelem, na który zostanie oparta aplikacja jest model architektury wielowarstwowej. Technologie, które są relatywnie nowe oraz warte są większej uwagi, zostaną wzbogacone o dodatkowy opis.

#### **3.1. Warstwa prezentacji**

Na rynku jest dostępne wiele platform do realizacji aplikacji mobilnych. Zaczynając od natywnych rozwiązań, w przypadku Android [33], Kotlin [42] oraz Java [48], przechodząc do IOS [7], czyli Objective-C [8] oraz Swift [9]. Wymienione rozwiązania nie będą jednak rozważane, gdyż jednym z założeń projektu jest zbudowanie aplikacji, która napisana jeden raz będzie mogła funkcjonować zarówno na urządzeniach z systemem Android, jak i z systemem IOS. Pomysł na takie podejście nie jest nowy. Jedną z pierwszych firm, która takie rozwiązanie zaprezentowała, był Facebook (2012 rok).

##### **3.1.1. React Native**

React Native [22] to trójplatformowa biblioteka, dostarczająca podprogramy służące programiście w rozwijaniu oprogramowania, bazowana na React [21]. Biblioteka ta używa JavaScript [10] lub TypeScript [46]. JavaScript, przyśpiesza rozwój aplikacji, jednak przy bardzo dużych projektach używanie go może prowadzić do błędów, wynikających z niezgodności typów, gdyż jest to język typowany dynamicznie, taki w którym typy zmiennych nie są jednoznacznie określane przez programistę. TypeScript jest zdefiniowany na podstawie JavaScript i wprowadza, w odróżnieniu do JavaScript, typizację statyczną, przeciwieństwo dynamicznej. Używanie typizacji statycznej w małych i średnich projektach wiąże się jednak ze znacznym wydłużeniem czasu potrzebnego na wyprodukowanie oprogramowania, co w większości przypadków nie jest potrzebne, a wręcz może być uciążliwe. React Native kompluluje się do native widgets [20], czyli komponentów natywnych. Są to specyficzne dla systemu jak i jego wersji komponenty takie jak, przyciski, formularze, paski aplikacji. Rozwiązanie to, nigdy nie będzie szybsze od całkowicie natywnego środowiska, Java (Android),

Objective-c (IOS), jednak pozwala na development na obydwu platformach, za pomocą jednego kodu.

Niesie to jednak ze sobą kilka wad:

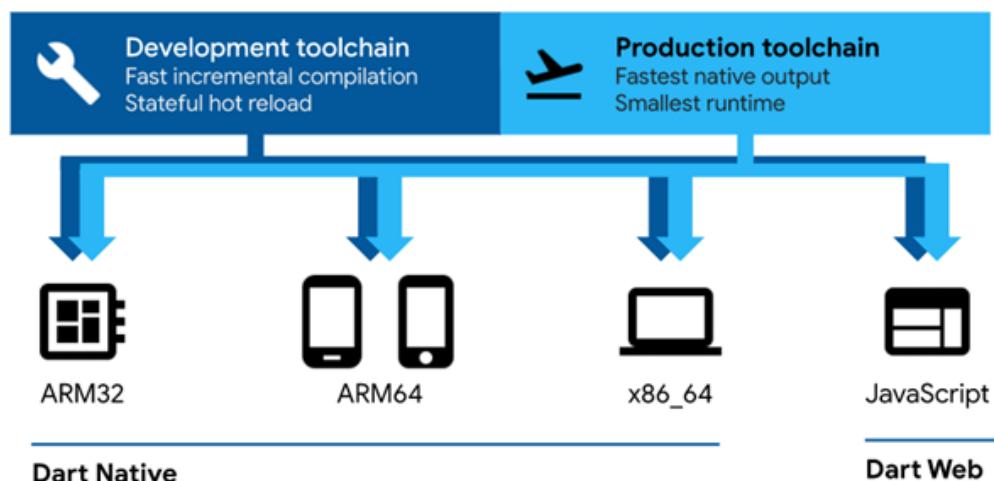
- aplikacja w zależności od systemu, jak i jego wersji, będzie wyglądać inaczej,
- przy aktualizacji systemu mogą pojawić się problemy związane z interfejsem użytkownika,
- w niektórych przypadkach, React Native wciąż potrzebuje wsparcia kodu natywnego, na przykład przy powiadomieniach *push*, czyli powiadomieniach "wypychanych" z serwera do aplikacji [67].

Z zalet można za to zauważyć:

- wspomniany wcześniej szybszy rozwój aplikacji na wiele platform,
- znajomość popularnego Reacta przyśpiesza pracę z React Native,
- praca z JavaScript.

### 3.1.2. Flutter

Flutter [40] to główny rywal React Native. Jest to wytworzony przez Google, zestaw narzędzi interfejsu użytkownika do tworzenia natywnie skompilowanych aplikacji. Nie tylko na urządzenia mobilne, ale również na strony internetowe oraz komputery z jednego i takiego samego kodu [41]. Flutter został wprowadzony w wersji 1.0 w 2018 roku. Mimo powyższych deklaracji firmy Google, na razie stabilne wersje, które są dostępne, to urządzenia mobilne oraz strony internetowe. Przy czym kompilowanie aplikacji do stron internetowych jest jeszcze w fazie beta. Aktualnie wersje skompilowane na komputery PC oraz MAC są dostępne w wersji alpha. Na wspomniane funkcje, zezwala fakt, że językiem programowania, którego używa Flutter jest Dart [36]. Możliwości Dartu przedstawione są na poniższym rysunku (rys. 3.1).



Rys. 3.1. Wygląd aplikacji Fixly dla wykonawców [37]

Kolejną znaczącą różnicą pomiędzy opisywanymi narzędziami, jest fakt, że Flutter jest natywny, a nie kompilowany do komponentów systemowych [41]. Flutter wyposażony jest w silnik graficzny Skia [41], który kontroluje każdy piksel na ekranie urządzenia. Takie rozwiązania daje ogromne możliwości w projektowaniu interfejsu użytkownika, przy niewielkiej stracie na optymalizacji [47]. Dodatkowo, aplikacja niezależnie od systemu operacyjnego oraz jego wersji będzie wyglądać tak samo na wszystkich urządzeniach, gdyż ta platforma spełnia już wymagania Material Design oraz Cupertino [41].

### 3.1.3. Ionic

Ionic [17] to framework *open source*, czyli taki który udostępnia kod źródłowy. Służy do tworzenia wydajnych, wysokiej jakości aplikacji mobilnych i stacjonarnych. Używając przy tym technologii internetowych - HTML, CSS i JavaScript. Jego najważniejszą cechą jest integracja z popularnymi narzędziami programistycznymi takimi jak Angular, React i Vue.[18]

Ionic koncentruje się na kontrolkach interfejsu użytkownika, gestach, animacjach. Alternatywnie można go używać samodzielnie, bez żadnego narzędzia używanego dla frontendu, warstwy prezentacji, używając prostego skryptu *include* [18]. Ionic jest dobrym wyborem, gdy developerzy są doświadczeni w platformach związanych z aplikacjami webowymi. Jednak nie zapewnia on takiej wydajności, jak opisane wcześniej narzędzia.

### 3.1.4. Podsumowanie

Po analizie zdecydowano, by w projekcie realizowanej w tej pracy, do warstwy prezentacji, użyć Fluttera. Jest to relatywnie nowa platforma programistyczna, w pełnej wersji dostępna od 2 lat. Wyposażona jest ona w nowoczesne funkcje, której wydajność porównywalna jest do rozwiązań natywnych dla wymienianych systemów operacyjnych (Android oraz IOS) [47].

## 3.2. Warstwa logiki

W projekcie zdecydowano się na podejście, w którym warstwę logiki przeniesiono do mikrousług oraz do systemu typu *headless CMS*, takiego którego podstawowymi składnikami są:

- baza danych,
- panel administracyjny,
- API.

Skorzystanie z samego headless CMS-a, prowadzi często do uproszczenia logiki biznesowej projektu, stąd też najczęściej używany jest on do małych, niewymagających projektów. Nowo tworzony system nie musi być jednak monolitem. Headless CMS, może zarządzać częścią danych, tych danych, które nie wymagają zaawansowanej

logiki biznesowej. Na przykład dodawaniem nowej usługi przez rzemieślnika. Bardziej zaawansowana logika, taka jak na przykład komuniukator, może zostać przeniesiona do mikrousług lub innych systemów. Takie innowacyjne podejście bardzo przyśpiesza pracę programisty. Dodatkowo, „bezgłowy” system do zarządzania danymi rozwiązuje wiele problemów związanych z własnym serwerem, takich jak skalowalność, utrzymanie oraz zarządzanie, a nie jest rozwiązaniem gorszym niż ręcznie skonfigurowana maszyna pod względem logiki biznesowej. Jedyną kwestię, którą należy rozważyć podczas używania *headless CMS*-a jest tworzenie kont użytkownika. Można to łatwo osiągnąć tworząc proste *proxy* (mikrousługi), uwierzytelniające, czyli funkcję bez serwerową przekazującą żądania klienta dalej, do *headless CMS*-a. Aktualnie najbardziej popularnym sposobem na obsługę opisanego przypadku w aplikacjach są rozwiązania szeroko rozumiane jako *Cloud Computing*. Większość, dzisiejszych aplikacji, nie potrzebuje serwera z pełnymi możliwościami, który obsługiwałby najdrobniejsze żądania, kierowane z warstwy prezentacji. Zaprogramowanie takiego urządzenia jest czasochłonne, a samo utrzymanie maszyny drogie. Stąd też usługi w warstwie logiki podzielono na *microservices*, mniejsze niezależne od siebie części systemu. Obsługiwane są one przez *serverless functions*. Są to funkcje wyposażone w endpointy, url do żądań HTTP, które zapewniają bezpośredni dostęp do funkcji i umieszczonych na nich *microservices*’ów. Aktualnym liderem rynku, jest firma Amazon [50], proponująca swoje rozwiązanie, nazwane AWS [2].

### 3.2.1. Amazon AWS

Rozwiązanie istnieje w ofercie AWS pod nazwą AWS Lambda [5]. Koszt obsługi *serverless functions* jest znikomy. W przypadku AWS jest to \$0,20 za 1 milion żądań. Przy czym warto zaznaczyć, że użytkownik AWS otrzymuje jeden milion żądań miesięcznie za darmo [6]. Dodatkowo AWS oferuje narzędzie do obsługi logowania i zarządzania użytkownikami, o nazwie AWS Cognito [4]. Serwis ten posiada duże możliwości ustawień parametrów, a próg wejścia w rozwiązanie jest dość wysoki. Szeroka gama konfiguracji wymaga jednak przez to o wiele więcej informacji do wdrożenia serwisu, może się to więc okazać większym problemem niż zaletą. Amazon również oferuje zbiór narzędzi integrujących aplikację z narzędziami proponowanymi w AWS. Amplify [3] został wdrożony do frameworka Flutter, we wczesnej wersji w sierpniu 2020 roku [54]. Rozwiązanie to wciąż może posiadać wady oraz błędy po stronie Amazona, jak i mało rozwiniętą społecznością. Z tych też powodów postanowiono nie brać tego rozwiązania pod uwagę.

### 3.2.2. Google Firebase

Firebase [38] to serwis cloud computingu proponowany przez firmę Google skupiony na aplikacjach mobilnych oraz platformach webowych. Firebase proponuje dużo ułatwień oraz uproszczeń w dziedzinie wdrażania swojej aplikacji na rynek w porównaniu do AWS. Specjalnie przygotowane paczki *SDK*, umożliwiają komunikację między warstwami aplikacji w kodzie, automatycznie wykonując oraz przyjmując żądania. Szczegółowa dokumentacja poparta licznymi przykładami oraz filmikami, ułatwia pracę z platformą. Paczka [25] utworzona specjalnie dla środowiska Flutter zdecydowanie przyspiesza rozwój aplikacji. Dodatkowo Firebase oferuje serwis, służący do obsługi kont użytkownika. Firebase Authentication [39] pozwala na wdrożenie funkcji logowania w mniej niż 10 linijkach kodu.[39]. Google przygotował specjalne elementy interfejsu użytkownika umożliwiające szybkie i proste wdrożenie logowania poprzez serwisy takie jak Google oraz Facebook [38]. W przyszłości ułatwi to rozszerzenie aplikacji o właśnie te funkcje.

Mimo tego, że Amazon AWS oraz Firebase proponują rozwiązania, bazodanowe typu NoSQL takie jak DynamoDB [1], a w przypadku Google, Cloud Firestore [35], to aby przyspieszyć proces tworzenia aplikacji, jak i zapewnić jej stabilność oraz skalowalność, do obsługi głównych danych aplikacji zostało wybrane rozwiązanie typu *headless CMS*. Rozwiązania NoSQL zostaną użyte do danych zmieniających się bardzo dynamicznie, czyli takich jak dane obsługujące komunikator. W przypadku dodawania ofert lub rejestracji, nie ma potrzeby na błyskawiczną odpowiedź systemu. W przypadku chatu taka potrzeba istnieje. Z tego też względu zdecydowano się na takie podejście.

### 3.2.3. Flotiq

Flotiq [12] to wypuszczona w 2020 platforma do zarządzania dowolnymi treściąmi. Po zdefiniowaniu *Content Type Definition*, Flotiq dostarcza użytkownikowi, API umożliwiające wysyłania do oraz pobierania z serwisu danych, wraz z szeroko pojętą dokumentacją. Oprócz tego Flotiq oferuje panel administracyjny (rys. 3.2) oraz udostępnia silnik wyszukiwania [15] oparty o Elastic Search [19]. Ze względu na fakt, że schemat danych we Flotiq jest zgodny z OpenAPI [61], użytkownik może wygenerować paczkę *SDK* [14] od obslugi danych, a następnie użyć jej w swojej aplikacji, bez potrzeby dodatkowej implementacji.

The screenshot shows the Flotiq administrative interface. On the left is a sidebar with a navigation menu:

- Dashboard
- Media library
- Content
  - authorprofile
  - Contact
  - craftsmen
  - masterpieces
- Product
  - service
  - test123
  - user
- Type definitions
- API doc

The main area is titled "Objects Of Type Product". It displays a table with the following data:

Id	Name	Slug	Price	Product image	Product gallery	Actions
product-2	Earl grey	earl-grey-tea	10			
product-4	Green tea	green-tea	15			
product-1	Wild fruit	wild-fruit	12			
product-3	English breakfast	english-breakfast	9			

At the bottom of the grid, there is a red button labeled "Remove selected (0)". Below the grid, there are pagination controls: "1 to 4 of 4", "Page 1 of 1", and "Reset grid".

Rys. 3.2. Panel administracyjny Flotiq [12]

### 3.2.4. Ghost

Ghost [28] to najbardziej popularny headless CMS na GitHub [55]. Jest to rozbudowana platforma, do publikowania treści, jednak są głównie pisane, na przykład w postaci artykułów. Główną zaletą Ghost jest szeroko rozbudowany edytor tekstu (rys. 3.3). Tekst pisany jest w formacie Markdown.

The screenshot shows the Ghost blog editor interface. On the left is a sidebar with navigation links: New Post, Content, Team, General, Navigation, Tags, Code Injection, Apps, and Labs. The main content area has a title "It's our birthday! Three years down, \$600,000 annual revenue, and what's coming next". The post content includes a paragraph about the author's return to a co-working space in Austria after launching Ghost, followed by a section titled "Ghost in Numbers: \$600,000 Annual Revenue" which contains a graph and a list of statistics. Below the graph is a summary paragraph and a note about finding financial data on Baremetrics. At the bottom right, it says "966 words".

On the 29th of April in 2013, I sat down in a beautiful little co-working space in Austria called [Daxbau](http://daxbau.at/), and launched the Ghost [Kickstarter campaign](https://www.kickstarter.com/projects/johnonolan/ghost-just-a-blogging-platform). Almost precisely 3 years later, I've returned to the exact same spot to write this post while I await the rest of the Ghost team to join me here for our first team trip of 2016.

It's always fun to take some time and look back at just how far we've come every 12 months, and I'm even more excited to tell you about where we'll be going next!

## Ghost in Numbers: \$600,000 Annual Revenue

The graph shows a steady upward trend in annual revenue over three years. The y-axis represents revenue in dollars, ranging from \$0 to \$60,000. The x-axis shows dates from Dec 14 to Apr 16, with labels for every two months. The revenue starts at approximately \$10,000 in Dec 14 and rises to about \$55,000 by Apr 16.

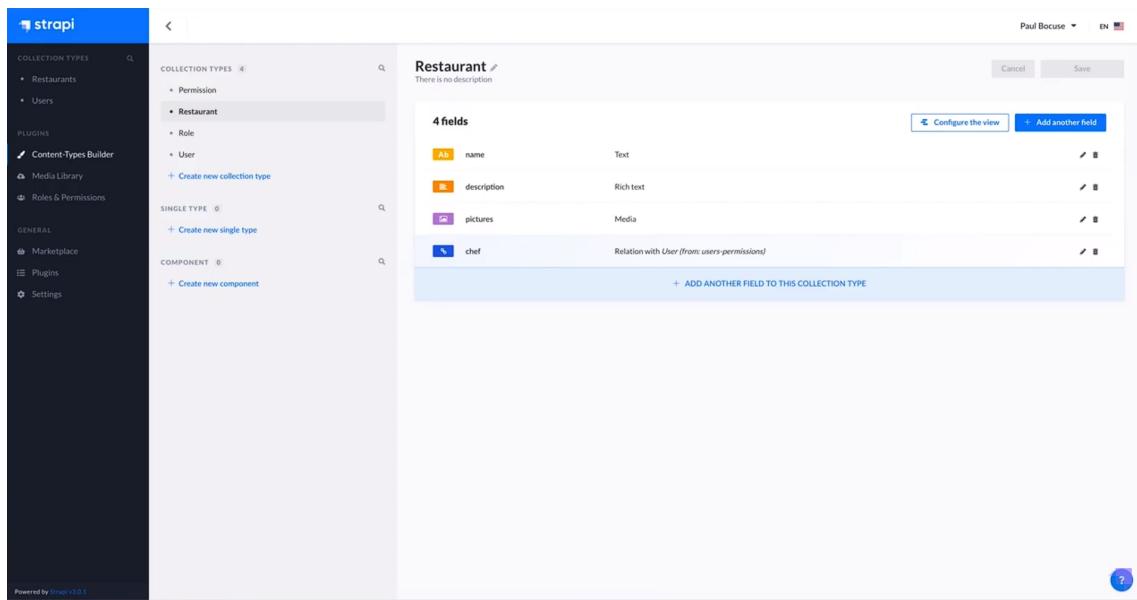
Date	Annual Recurring Revenue (ARR)
Dec 14	\$10,000
Feb 15	\$25,000
Apr 16	\$55,000

- Total downloads: 1,088,580
- Github Stars: 19,301
- Ghost(Pro) subscribers: 4,705
- Monthly recurring revenue (MRR): \$50,038
- Annual recurring revenue (ARR): \$600,456
- Average revenue per user: \$10.59
- Churn rate: 4.9%
- Team size: 9 people across 5 countries and 3 continents (+2 on trial, and 2 open positions)

Rys. 3.3. Edytor Markdown w Ghost [28]

### 3.2.5. Strapi

Strapi [63] to drugi, po Ghost, najbardziej popularny „bezgłowy” system do zarządzania danymi [55]. W Strapi użytkownik, definiuje własne *Content Type Definition* (rys. 3.4). Następnie generowane jest *API*, które pozwala na dostęp do treści, jak i jej modyfikację. Strapi posiada również możliwość wyszukiwania treści umieszczonych w systemie [64]. Najważniejszym atutem Strapii jest fakt, że posiada ono własną metodę autentykacji, czyli rejestracji oraz logowania użytkowników do systemu.



Rys. 3.4. Dodawanie *CTD* w strapi [63]

### 3.3. Użyte technologie

Przedstawiona w powyższych podrozdziałach analiza pozwoliła na decyzję o tym, że warstwa logiki w projekcie składać będzie się z:

- Firebase Cloud Functions,
- Firebase Cloud Firestore,
- Firebase Authentication,
- Flotiq.

Zdecydowano się na użycie rozwiązań z rodzinny Firebase z dwóch głównych powodów. Pierwszym jest to, że integracja warstwy prezentacji z warstwą logiki, będzie szczególnie prosta, gdyż Flutter oraz Firebase są rozwijane przez Google. Oznacza to, że są przygotowane tak by ułatwić deweloperom łączenie tych technologii. Drugim powodem jest prostota konfiguracji serwisu Firebase. AWS mimo dużych możliwości konfiguracji, wymaga też definicji o wiele większej liczby parametrów,

by móc rozpocząć pracę z serwisem. W przypadku Firebase rozpoczęcie pracy z narzędziem, zaczyna się od razu po założeniu projektu, cechuje się on też zdecydowanie krótszym *time to market*, czyli czasem wdrożenia aplikacji na rynek, niż AWS. Przy czym, określenie *time to market* nie odnosi się bezpośrednio do publikacji aplikacji, a do czasu, który jest potrzebny, by aplikacja była w pełni gotowa do użytku. Wsparcie ze strony Google Developers, jak i ze strony społeczności utwierdza tylko w tej decyzji.

Z opisanych platform zarządzania treścią do projektu wybrano Flotiq. Ghost jest platformą przewidzianą głównie do prowadzenia blogów i stron z artykułami. W przypadku Strapi można zarządzać szeroko pojmowaną treścią, jednak w przeciwieństwie do Flotiq, nie posiada on tak rozbudowanego silnika wyszukiwania oraz nie umożliwia wygenerowania paczki *SDK*. Obydwa odrzucone rozwiązania wymagają osobnego, skonfigurowanego przez dewelopera, urządzenia, by mogły funkcjonować. Flotiq dostępny jest pod stałym adresem i zarządzany jest przez jego twórców. Fakt ten jest o tyle ważnym, że w przypadku Strapii, cała odpowiedzialność związana z bezpieczeństwem serwera, jego utrzymaniem przerzucona jest na dewelopera. Bezpieczeństwo rozwiązania jest najważniejsze, gdyż taki serwer będzie trzymać poufne dane użytkowników. Najmniejsze zaniedbanie związane z zabezpieczeniami prowadzić może do wycieku danych a co za tym idzie konsekwencji prawnych. Przedstawiony argument, tym bardziej utwierdza w podjętej decyzji.

Do zabezpieczenia API przygotowanego przez Flotiq zdecydowano się użyć Firebase Cloud Functions połączonych z Firebase Authentication. *Serverless functions* służąć będą jako swego rodzaju proxy uwierzytelniające. Paczki przygotowane przez Firebase umożliwiają na wywoływanie funkcji z kodu warstwy prezentacji. W przypadku AWS nie byłoby to możliwe.

### 3.4. Narzędzia pomocnicze

Wybór narzędzi pomocniczych nie wpływa bezpośrednio na sam kod aplikacji, jednak może znacząco wpłynąć na samą pracę programisty. Sekcję tę podzielono na kilka najważniejszych narzędzi:

- system kontroli wersji,
- IDE, zintegrowane narzędzie programistyczne do pisania kodu,
- system operacyjny komputera,
- emulator telefonu,
- narzędzie do prototypowania.

#### 3.4.1. System kontroli wersji

Aktualnie najpopularniejszym narzędziem do kontroli wersji jest Git [65]. Git [51] to darmowe oraz otwarte źródłowo narzędzie, posiadające ogromne wsparcie społeczności. Dodatkowo jest bezpośrednio wspierane w wielu IDE. Kolejnym aspektem jest

narzędzie do przechowywania kodu źródłowego w chmurze. Wybór ten nie ma dużego znaczenia, z powodu braku zespołu pracującego nad aplikacją. Stąd też zdecydowano na użycie serwisu GitHub [29]. Stworzono w nim prywatne repozytorium, którego zadaniem jest trzymanie, w chmurze, aktualnego kodu aplikacji, a w razie potrzeby dostęp do niego z innego urządzenia.

### 3.4.2. IDE

Za wspomaganie pisania kodu programiście, odpowiedzialne są dwa pluginy, czyli dodatki do programu głównego, stworzone przez osoby trzecie. Są to *Dart* oraz *Flutter*. Pluginy te dostępne są na wiele IDE, stąd też inne aspekty mają większa znaczenie. Między innymi, obsługa gita, responsywność oraz szybkość uruchamiania się samego IDE. Pod uwagę zostały wzięte 3 główne narzędzia:

- Visual Studio Code [53],
- Android Studio [34],
- IntelliJ IDEA Ultimate [49].

IntelliJ IDEA Ultimate jest jedynym płatnym narzędziem, jednak z powodu statusu studenta autora, jest ono dostępne za darmo. Android Studio jest narzędziem propozowanym przez Google do pracy nad Flutterem, posiada ono również dobry wsparcie dla gita. Jednak z racji tego, że wymaga ono dużo czasu na uruchomienia oraz często sprawia problemy przy aktualizacjach samego IDE, odrzucono to narzędzie. Visual Studio Code, to narzędzie, które uruchamia się najszybciej ze wszystkich podanych, jak również jest najbardziej responsywne. Jedyną zaś znaczącą wadą tego IDE jest brak dobrego wsparcia dla gita. IntelliJ IDEA Ultimate jest narzędziem łączące cechy dwóch opisanych wcześniej IDE. Posiada dobre wsparcie dla gita oraz uruchamia się szybciej niż android studio. Z tych też względów zdecydowano się na użycie tego narzędzia.

### 3.4.3. System operacyjny komputera

Wybór systemu operacyjnego w przypadku Fluttera jest istotny. Firma Apple nie udostępnia emulatorów swojego urządzenia na inne systemy niż macOS. Stąd też najlepszym wyborem, do pracy nad aplikacją, byłby komputer ze wspomnianym oprogramowaniem. Pozwala to na emulowanie oraz rozwijanie aplikacji na systemie Android, jak i na systemie IOS. Dużym ograniczeniem takiego rozwiązania jest jego koszt. Urządzenia z systemem macOS, nawet o gorszych specyfikacjach, są bardzo drogie w porównaniu do urządzeń z systemem Windows. Z tego też powodu w pracy nie znajdę się rysunki ukazujące uruchomienie aplikacji na systemie IOS, gdyż autor nie ma takiej możliwości.

#### **3.4.4. Emulator urządzenia**

Wybór emulatora miałby znaczenie w przypadku dostępu do urządzenie z systemem macOS. Jeżeli chodzi zaś o system Windows, wybrano Android Emulator [44] autorstwa Google. Emulator można uruchamiać z konsoli systemowej, a dodatkowo wybrane IDE, ma do tego dedykowany przycisk.

#### **3.4.5. Narzędzie do prototypowania**

Aktualnie dwoma najbardziej popularnymi oraz darmowymi narzędziami są Figma oraz Adobe XD [66]. Figma [24] to narzędzie do tworzenia prototypów interfejsu użytkownika. Narzędzie to dostępne jest w przeglądarce, a wszystkie projekty użytkownika zapisywane są automatycznie w chmurze. Adobe XD to również narzędzie do tworzenia prototypów interfejsu użytkownika. Nie jest jednak dostępne w przeglądarce, a by używać go lokalnie bezpośrednio na urządzeniu, potrzebne jest dodatkowe, również darmowe, oprogramowanie Creative Cloud. W przypadku Figmy, nie ma potrzeby instalowania dodatkowych programów [23]. Z tego powodu, jak i również możliwości dostępu do aplikacji przez przeglądarkę, zdecydowano na użycie Figmy.

### **3.5. Podsumowanie**

Przegląd technologii dostępnych na rynku pozwolił na wybór tych, które zapewnią optymalizację aplikacji, przy mniejszym nakładzie pracy potrzebnym do wykonania takiego projektu. Dzięki temu w krótszym czasie zostanie wytworzone oprogramowanie, z większą liczbą funkcji, a utrzymanie projektu po jego wdrożeniu będzie dużo prostsze oraz tańsze.

## **4. Projekt**

### **4.1. Wymagania projektowe**

Projekt powstaje w celu wytworzenia aplikacji mobilnej wspomagającej komunikację rzemieślnika z użytkownikiem jego wyrobów, a co za tym idzie wypełnienie luki istniejącej na rynku, opisanej w rozdziale drugim pracy. Dodatkowym aspektem jest promocja kultury rzemieślników miejskich. Przekłada się to na następujące funkcje, które zostały zdefiniowane również w rozdziale drugim:

- tworzenie ofert,
- personalizacja sklepu,
- system ocen,
- chat,
- mapa z zaznaczonymi warsztatami rzemieślniczymi,
- wyszukiwarka po ofertach oraz rzemieślnikach,
- lokalizacja pobliskich warsztatów rzemieślniczych.

Grupą docelową aplikacji są osoby posiadające urządzenie mobilne z systemem operacyjnym Andorid, od wersji 6.0 oraz IOS, od wersji 8. Wspomniane wersje systemu są o tyle ważne, że w przypadku Android wersja systemu od 6.0 wzwyż pokrywa 91% całego rynku [62] telefonów oraz tabletów z systemem Android, w przypadku IOS od wersji 10.0. Pierwszą podgrupą są użytkownicy, potrzebujący jakiejś usługi bądź też produktu. Drugą podgrupą są rzemieślnicy, świadczący usługi oraz sprzedający wykonane przez siebie przedmioty. Po uruchomieniu aplikacji, powinien wyświetlać się ekran, który pozwala wybrać, czy osoba zainteresowana chce zalogować się jako użytkownik czy jako rzemieślnik. Użytkownikiem w aplikacji może być każdy. Informacja, jaka powinna być potrzebna do utworzenia konta, to adres e-mail. Po zalogowaniu się, odbiorca będzie mógł:

- wyszukać na mapie pobliskich mu rzemieślników,
- wyszukać interesujące go przedmioty,
- przeglądać przedmioty oraz usługi konkretnego rzemieślnika,
- rozpocząć chat z wybranym rzemieślnikiem,
- znać status swojego zamówienia,
- ocenić rzemieślnika,
- skonfigurować własny profil.

Aplikacja zorientowana ma być na rzemieślników, jednak jej ważniejszym elementem są użytkownicy. Pokazała to analiza rynku omówiona przedstawiona w rozdziale drugim niniejszej pracy. Duże zainteresowanie tylko ze strony usługodawców nie świadczy o sukcesie aplikacji. Stąd też największą wagę należy nadać części aplikacji poświęconej usługobiorcy.

Rzemieślnik do założenia konta w aplikacji, powinien podać swoje imię i nazwisko, lokalizację sklepu oraz nazwę swojej działalności, a także nazwę samego warsztatu. Po zalogowaniu się usługodawca będzie mógł:

- dodawać zdjęcia do profilu swojego warsztatu,
- personalizować profil swojego warsztatu,
- odpowiadać na chat użytkownika,
- dodawać nowe oferty,
- personalizować swój profil,
- dodawać usługi, produkty oraz produkty na zamówienie.

Najważniejszym wskaźnikiem związanym z popularnością będzie liczba użytkowników. Jeżeli w przeciągu roku, od hipotetycznego wprowadzenia aplikacji na rynek, liczba odbiorców przekroczy 50 tysięcy, projekt będzie można uznać za sukces.

## 4.2. Wymagania funkcjonalne

Wymagania funkcjonalne zostały przedstawione w postaci historyjek użytkownika.

**WY-01** Jako gość chcę mieć możliwość rejestracji, jako rzemieślnik oraz jako użytkownik.

**WY-02** Jako gość chcę mieć możliwość logowania się na swoje konto.

**WY-03** Jako rzemieślnik chcę mieć możliwość dostosowania swojego profilu.

**WY-04** Jako rzemieślnik chcę mieć możliwość dostosowania swojego warsztatu.

**WY-05** Jako rzemieślnik chcę mieć możliwość dodawania, usuwania, zmiany ofert na wykonywane przeze mnie przedmioty.

**WY-06** Jako rzemieślnik chcę mieć możliwość dodawania, usuwania, zmianiania usług w swoim profilu.

**WY-07** Jako rzemieślnik chcę mieć możliwość wysyłania zdjęć oraz wiadomości tekstowych do użytkowników.

**WY-08** Jako rzemieślnik chcę mieć możliwość wylogowania się z aplikacji.

**WY-09** Jako rzemieślnik chcę mieć możliwość przeglądania swojego profilu oraz warsztatu.

**WY-10** Jako rzemieślnik chcę mieć możliwość obsługi zamówienia.

**WY-11** Jako użytkownik chcę mieć możliwość logowania się na swoje konto.

**WY-12** Jako użytkownik chcę mieć możliwość przeglądania rzemieślników.

- WY-13** Jako użytkownik chcę mieć możliwość przeglądania ofert oraz usług rzemieślników.
- WY-14** Jako użytkownik chcę mieć możliwość wyszukania na mapie położenia warsztatu rzemieślnika.
- WY-15** Jako użytkownik chcę mieć możliwość przeglądania warsztatów rzemieślniczych w mojej okolicy.
- WY-16** Jako użytkownik chcę mieć możliwość wysyłania wiadomości tekstowych oraz zdjęć do rzemieślnika.
- WY-17** Jako użytkownik chcę mieć możliwość dodania oraz usunięcia rzemieślnika z obserwowanych.
- WY-18** Jako użytkownik chcę mieć możliwość dodania oraz usunięcia produktu z obserwowanych.
- WY-19** Jako użytkownik chcę mieć możliwość wystawienia oceny po wykonanym zamówieniu przez rzemieślnika.
- WY-20** Jako użytkownik chcę mieć możliwość wylogowania się ze swojego konta.
- WY-21** Jako użytkownik chcę mieć możliwość przeglądania swojego profilu.

#### **4.3. Wymagania niefunkcjonalne**

- WY-24** Aplikacja powinna uruchamiać się w mniej niż 5 sekund.
- WY-25** Aplikacja powinna być responsywna – użytkownik nie powinien czekać dłużej na reakcję aplikacji niż 5 sekund.
- WY-26** Aplikacja powinna funkcjonować w języku polskim.
- WY-27** Aplikacja powinna funkcjonować na systemie Android od wersji 6.0 a na systemie IOS od wersji 10.0.
- WY-28** Do poprawnego funkcjonowania mapy potrzebne jest zezwolenie na użycie lokalizacji.
- WY-29** Do poprawnego funkcjonowania aplikacji potrzebne jest połączenie z Internetem.
- WY-30** Aplikacja dla nowego użytkownika powinna być intuicyjna, a nauka jej obsługi powinna zająć mniej niż 15 minut.

#### **4.4. Model przypadków użycia**

Model przypadków użycia składa się z:

- opisu tekstuowego aktorów i usług,
- zbioru diagramów,
- szczegółowego opisu każdego przypadku użycia, czyli scenariuszy przypadków użycia.

Przypadek użycia opisuje poniższa definicja na postawie [16].

Przypadek użycia przedstawia to co system może zaoferować swoim użytkownikom. Są to funkcje systemu, które są widziane z zewnątrz. Przypadki użycia połączone z innymi czynnościami tworzą proces, który płynnie scala interfejs użytkownika z podstawowymi celami systemu [16].

Opisywany projekt, został opatrzony w przypadki użycia. Opierają się one o przedstawione w punkcie 4.1 wymagania funkcjonalne. Każdy przypadek użycia opisuje funkcję zawartą w opisywanej aplikacji, a dodatkowo, by lepiej przedstawić cały model, został on zwarty w diagram przypadków użycia oraz uszczegółowiony w scenariuszach przypadków użycia umiejscowionych w podrozdziałach tego rozdziału.

## Spis Przypadeków Użycia

- PU-01** Wylogowanie
- PU-02** Logowanie
- PU-03** Rejestracja
- PU-04** Konfiguracja profilu
- PU-05** Konfiguracja warsztatu rzemieślnika
- PU-06** Przeglądanie własnego profilu
- PU-07** Przeglądanie profilu rzemieślnika
- PU-08** Przeglądanie warsztatu rzemieślnika
- PU-09** Chat z rzemieślnikiem
- PU-10** Chat z użytkownikiem
- PU-11** Dodawanie ofert i usług
- PU-12** Wyszukiwanie ofert i usług lub rzemieślników
- PU-13** Składanie zamówienia
- PU-14** Obsługa zamówień
- PU-15** Lokalizacja oraz przeglądanie mapy
- PU-16** Dodanie przedmiotu do obserwowanych
- PU-17** Dodanie rzemieślnika do obserwowanych

*Dla utrzymania prostoty Spisu Przypadeków Użycia jak i diagramu przypadków użycia w **PU-14** zostało zawarte kilka akcji związanych z zarządzaniem zamówieniem przez rzemieślnika oraz użytkownika. Dokładny spis akcji znajduje się nad tabelą 4.14. tabela ta przedstawia scenariusz tego przypadku użycia. Dodatkowo w **PU-12** pojęcie oferta odnosi się całkowicie do przedmiotów wystawianych na sprzedaż przez rzemieślnika, czy to na zamówienie, czy też dostępnych od ręki.*

### 4.4.1. Diagram przypadków użycia

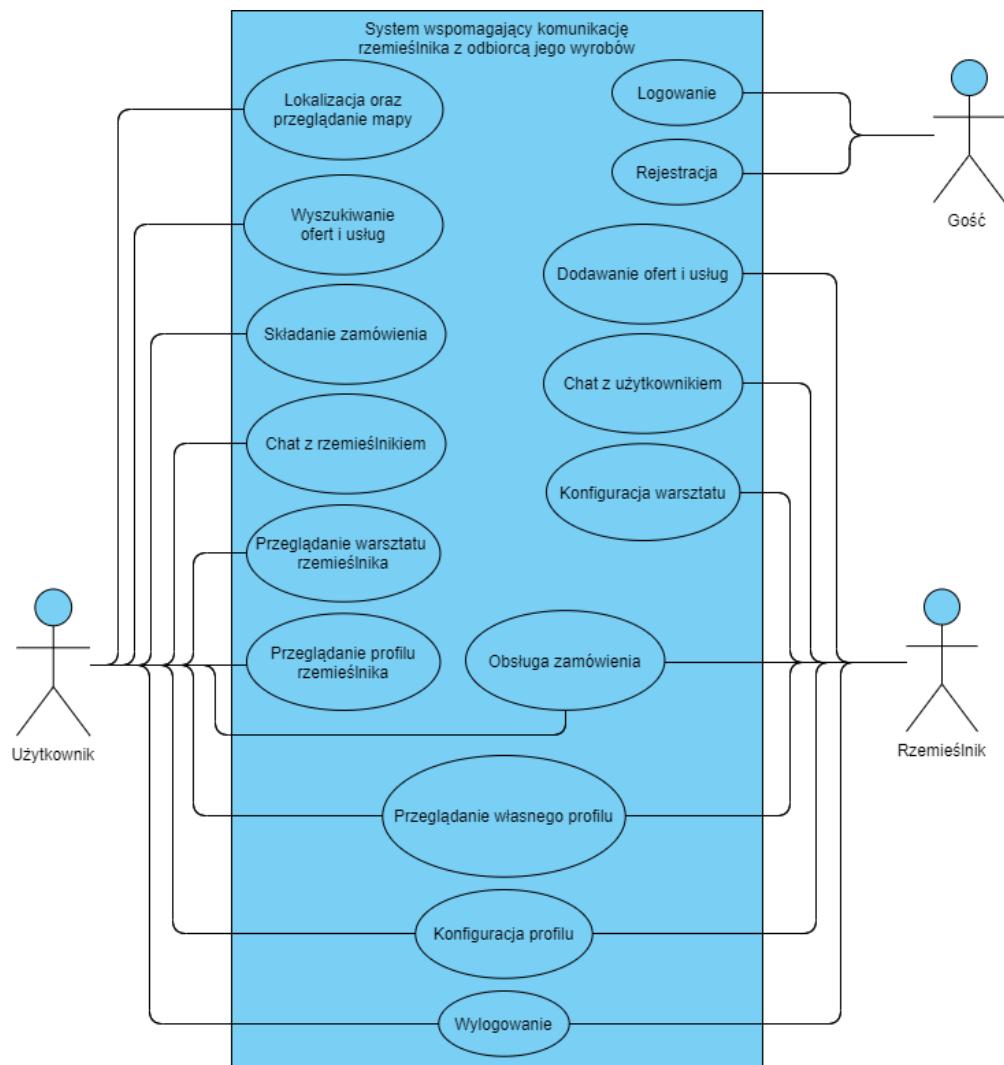
Na podstawie definicji przedstawionej przez Cockburna w [11]. Diagram przypadków (PU) użycia to:

Zbiór scenariuszy, które mogą wystąpić pomiędzy systemem, a aktorami spoza systemu. Scenariusze te charakteryzowane są przez cel, jaki aktor ma wobec systemu, w stosunku do zadeklarowanych przez system obowiązków. Pokazuje, jak podstawowy cel aktora może zostać osiągnięty lub może się nie powieść [11]

Poniższy rys. 4.1 przedstawia diagram przypadków użycia. Przedstawieni aktorzy to:

- Gość,
- Użytkownik,
- Rzemieślnik.

Przy czym **Gość** to osoba, która po logowaniu (**PU-02**), bądź też rejestracji (**PU-01**) może stać się **Rzemieślnikiem** lub **Użytkownikiem**. Wszyscy trzej aktorzy wykonują przypadki, z którymi są połączeni.



Rys. 4.1. Diagram przypadków użycia [źródło: opracowanie własne]

#### 4.4.2. Scenariusze przypadków użycia

Jako uszczegółowienie diagramu (Rys. 4.1), stosuje się scenariusze przypadków użycia. Aby dobrze przedstawić czym one dokładnie są ponownie przytoczono [11]. Scenariusz PU to:

sekwencja interakcji, które zachodzą przy określonych warunkach wejściowych, osiągając cel główny aktora, czyli określony rezultat w odniesieniu do tego celu. Cel ten odnosi się bezpośrednio do warunków wyjściowych. Sekwencja rozpoczyna się od działania wyzwalającego i trwa dopóki cel zostanie osiągnięty lub porzucony. System w odpowiedzi na interakcję wypełnia wszystkie zadeklarowane przez siebie obowiązki. [11]

Poniżej, w tabelach od 4.1 do 4.17, przedstawiono scenariusze przypadków użycia. Pomiędzy tabelami znajdują się dodatkowe objaśnienia odnoszące się do konkretnych scenariuszy PU.

Zabezpieczenia konta użytkownika zostały zawarte w 3 pierwszych scenariuszach tabela 4.3 (Rejestracja), tabela 4.2 (Logowanie) oraz tabela 4.1 (Wylogowanie). Gość może się zarejestrować oraz zalogować. Rzemieślnik oraz użytkownik może się wylogować.

Tabela 4.1. Scenariusz PU-01 - Wylogowanie

<b>Przypadek użycia</b>	PU-01
<b>Nazwa przypadku</b>	Wylogowanie.
<b>Cel</b>	Wylogowanie się z konta.
<b>Aktorzy</b>	Użytkownik, Rzemieślnik.
<b>Warunki wejściowe</b>	Użytkownik lub rzemieślnik jest zalogowany.
<b>Warunki wyjściowe</b>	Użytkownik lub rzemieślnik jest wylogowany, zamienia się w gościa.
<b>Scenariusze</b>	
<b>Przebieg główny:</b>	<ol style="list-style-type: none"><li>Użytkownik lub rzemieślnik, naciska przycisk <i>Wyloguj.</i></li><li>System wylogowuje użytkownika.</li><li>System przenosi użytkownika do strony logowania oraz informuje o poprawnym wylogowaniu.</li></ol>

Tabela 4.2. Scenariusz PU-02 - Logowanie

<b>Przypadek użycia</b>	PU-02
<b>Nazwa przypadku</b>	Logowanie.
<b>Cel</b>	Uwierzytelnienie gościa w celu udostępnienia mu odpowiednich treści oraz funkcji.
<b>Aktorzy</b>	Gość.
<b>Warunki wejściowe</b>	Gość nie jest zalogowany.
<b>Warunki wyjściowe</b>	Gość posiada dostęp do konta, stając się użytkownikiem lub rzemieślnikiem.
<b>Scenariusze</b>	
<b>Przebieg główny:</b> <i>Dane nie są poprawne.</i>	<ol style="list-style-type: none"> <li>1. Gość podaje e-mail oraz hasło.</li> <li>2. Gość naciska przycisk <i>Zaloguj się</i>.</li> <li>3. System sprawdza poprawność danych.</li> <li>4. System potwierdza poprawność danych.</li> <li>5. System przenosi gościa do panelu użytkownika.</li> </ol>
<b>Przebieg alternatywny:</b> <i>Dane nie są poprawne.</i>	<p>4A-1. System informuje gościa o niepoprawnych danych.</p> <p>4A-2. Gość poprawia niepoprawne dane wskazane, przez system.</p> <p>4A-3. Powrót do punktu 2.</p>

Tabela 4.3. Scenariusz PU-03 - Rejestracja

<b>Przypadek użycia</b>	PU-03
<b>Nazwa przypadku</b>	Rejestracja.
<b>Cel</b>	Utworzenie nowego użytkownika w systemie, w celu umożliwienia mu korzystania z aplikacji.
<b>Aktorzy</b>	Gość.
<b>Warunki wejściowe</b>	Użytkownik nie jest zalogowany.
<b>Warunki wyjściowe</b>	Użytkownik posiada konto w aplikacji.
<b>Scenariusze</b>	
<b>Przebieg główny:</b> <i>Dane nie spełniają wymagań.</i>	<ol style="list-style-type: none"> <li>1. Gość podaje e-mail oraz hasło dwa razy.</li> <li>2. Gość naciska przycisk <i>Zarejestruj się</i>.</li> <li>3. System sprawdza poprawność danych.</li> <li>4. System potwierdza poprawność danych.</li> <li>5. System dodaje użytkownika do bazy danych.</li> <li>6. System przenosi gościa do panelu użytkownika.</li> </ol>
<b>Przebieg alternatywny:</b> <i>Użytkownik z podanym adresem e-mail już istnieje.</i>	<p>4A-1. System informuje gościa o niepoprawnych danych.</p> <p>4A-2. Gość poprawia dane wskazane, przez system.</p> <p>4A-3. Powrót do punktu 2.</p> <p>4B-1. System informuje gościa o niepoprawnym e-mailu.</p> <p>4B-2. Gość wprowadza e-mail, który nie widnieje w bazie danych.</p> <p>4B-3. Powrót do punktu 2.</p>

Użytkownik oraz rzemieślnik po rejestracji może skonfigurować swój profil. Przykład ten opisuje tabela 4.4.

Tabela 4.4. Scenariusz PU-04 - Konfiguracja profilu

<b>Przypadek użycia</b>	PU-04
<b>Nazwa przypadku</b>	Konfiguracja profilu.
<b>Cel</b>	Dostosowanie własnego profilu.
<b>Aktorzy</b>	Użytkownik, Rzemieślnik.
<b>Warunki wejściowe</b>	Aktor jest zalogowany.
<b>Warunki wyjściowe</b>	Nowe dane aktora zapisane są w systemie.
<b>Scenariusze</b>	
<b>Przebieg główny:</b>	<ol style="list-style-type: none"> <li>1. Aktor wprowadza swoje dane.</li> <li>2. Aktor dodaje swoje zdjęcie.</li> <li>3. Aktor naciska przycisk <i>Zapisz</i>.</li> <li>4. System sprawdza poprawność danych.</li> <li>5. System potwierdza poprawność danych.</li> <li>6. System zapisuje dane.</li> <li>7. System informuje o tym, że dane zostały zapisane.</li> </ol>
<b>Przebieg alternatywny:</b> <i>Dane nie spełniają wymagań.</i>	<p>5A-1. System informuje aktora o niepoprawnych danych.</p> <p>5A-2. Aktor poprawia dane wskazane, przez system.</p> <p>5A-3. Powrót do punktu 3.</p>
<b>Przebieg alternatywny:</b> <i>Aktor dodaje i zmienia rozmiar zdjęcia.</i>	<p>2A-1. Aktor naciska przycisk <i>Modyfikuj</i>.</p> <p>2A-2. Aktor modyfikuje zdjęcie.</p> <p>2A-3. Aktor naciska przycisk <i>Zapisz zmiany</i>.</p> <p>2A-4. Powrót do punktu 3.</p>

Rzemieślnik po rejestracji dodatkowo, może skonfigurować swój warsztat, dodając zdjęcie, opis, lokalizację oraz nazwę. Scenariusz ten opisany jest w tabeli 4.5.

Tabela 4.5. Scenariusz PU-05 - Konfiguracja warsztatu rzemieślnika

<b>Przypadek użycia</b>	PU-05
<b>Nazwa przypadku</b>	Konfiguracja warsztatu rzemieślnika.
<b>Cel</b>	Dostosowanie własnego warsztatu.
<b>Aktorzy</b>	Rzemieślnik.
<b>Warunki wejściowe</b>	Rzemieślnik jest zalogowany.
<b>Warunki wyjściowe</b>	Nowe dane o warsztacie zapisane są w systemie.
<b>Scenariusze</b>	
<b>Przebieg główny:</b> <i>Dane nie spełniają wymagań.</i>	<ol style="list-style-type: none"> <li>1. Rzemieślnik wprowadza nowe dane o warsztacie.</li> <li>2. Rzemieślnik dodaje zdjęcie.</li> <li>3. Rzemieślnik naciska przycisk <i>Zapisz</i>.</li> <li>4. System sprawdza poprawność danych.</li> <li>5. System potwierdza poprawność danych.</li> <li>6. System zapisuje dane.</li> <li>7. System informuje o tym, że dane zostały zapisane.</li> </ol>
<b>Przebieg alternatywny:</b> <i>Rzemieślnik dodaje i zmienia rozmiar zdjęcia.</i>	<p>5A-1. System informuje rzemieślnika o niepoprawnych danych.</p> <p>5A-2. Rzemieślnik poprawia dane wskazane, przez system.</p> <p>5A-3. Powrót do punktu 3.</p>
<b>Przebieg alternatywny:</b> <i>Rzemieślnik naciska przycisk Modyfikuj.</i>	<p>2A-1. Rzemieślnik naciska przycisk <i>Modyfikuj</i>.</p> <p>2A-2. Rzemieślnik modyfikuje zdjęcie.</p> <p>2A-3. Rzemieślnik naciska przycisk <i>Zapisz zmiany</i>.</p> <p>2A-4. Powrót do punktu 3.</p>

Tabela 4.6. Scenariusz PU-06 - Przegląd własnego profilu

<b>Przypadek użycia</b>	PU-06
<b>Nazwa przypadku</b>	Przegląd własnego profilu.
<b>Cel</b>	Sprawdzenie jak widziany jest profil przez innych użytkowników lub aktorów.
<b>Aktorzy</b>	Użytkownik, rzemieślnik.
<b>Warunki wejściowe</b>	Użytkownik lub rzemieślnik jest zalogowany.
<b>Warunki wyjściowe</b>	Użytkownik wie jak rzemieślnicy widzą jego profil. Rzemieślnik wie jak użytkownicy widzą jego profil.
<b>Scenariusze</b>	
<b>Przebieg główny:</b>	<ol style="list-style-type: none"> <li>1. Aktor naciska przycisk <i>Wyświetl profil</i>.</li> <li>2. System wyświetla profil.</li> </ol>

W profilu rzemieślnika można znaleźć dużo informacji. Między innymi oferty oraz usługi wystawione przez rzemieślnika. Przegląd profilu rzemieślnika przedstawiono w tabeli 4.7.

Tabela 4.7. Scenariusz PU-07 - Przeglądanie profilu rzemieślnika

<b>Przypadek użycia</b>	PU-07
<b>Nazwa przypadku</b>	Przeglądanie profilu rzemieślnika.
<b>Cel</b>	Sprawdzenie oferty rzemieślnika.
<b>Aktorzy</b>	Użytkownik.
<b>Warunki wejściowe</b>	Brak.
<b>Warunki wyjściowe</b>	Użytkownik zapoznał się z ofertą rzemieślnika.
<b>Scenariusze</b>	
<b>Przebieg główny:</b>	<ol style="list-style-type: none"> <li>1. Użytkownik wybiera rzemieślnika.</li> <li>2. Użytkownik naciska na kafelek opisującego rzemieślnika.</li> <li>3. System wyświetla profil rzemieślnika.</li> </ol>

Informacje takie jak lokalizacja oraz godziny otwarcia znajdują się w warsztacie rzemieślnika. Przypadek PU-08 Przeglądanie warsztatu rzemieślnika opisany jest poniżej w tabeli 4.8.

Tabela 4.8. Scenariusz PU-08 - Przeglądanie warsztatu rzemieślnika

<b>Przypadek użycia</b>	PU-08
<b>Nazwa przypadku</b>	Przeglądanie warsztatu rzemieślnika.
<b>Cel</b>	Sprawdzenie warsztatu rzemieślnika.
<b>Aktorzy</b>	Użytkownik.
<b>Warunki wejściowe</b>	Wykonuje PU-07 Przeglądanie warsztatu rzemieślnika.
<b>Warunki wyjściowe</b>	Użytkownik zapoznał się z warsztatem rzemieślnika.
<b>Scenariusze</b>	
<b>Przebieg główny:</b>	<ol style="list-style-type: none"> <li>1. Użytkownik naciska na przycisk <i>Warsztat</i>.</li> <li>2. System wyświetla profil rzemieślnika.</li> </ol>

Waczną funkcją aplikacji jest czat użytkownika z rzemieślnikiem. Przypadek użycia opisany jest w tabeli 4.9

Tabela 4.9. Scenariusz PU-09 - Chat z rzemieślnikiem

<b>Przypadek użycia</b>	PU-09
<b>Nazwa przypadku</b>	Chat z rzemieślnikiem
<b>Cel</b>	Chat z rzemieślnikiem.
<b>Aktorzy</b>	Użytkownik
<b>Warunki wejściowe</b>	Użytkownik wykonuje PU-07
<b>Warunki wyjściowe</b>	Użytkownik wysłał wiadomość do rzemieślnika
<b>Scenariusze</b>	
<b>Przebieg główny:</b>	<ol style="list-style-type: none"> <li>1. Użytkownik naciska na ikonę z dymkiem.</li> <li>2. System przenosi użytkownika do ekranu chatu z wybranym rzemieślnikiem.</li> <li>3. Użytkownik wpisuje wiadomość.</li> <li>4. Użytkownik naciska ikonę strzałki.</li> <li>5. System zapisuje wiadomość w systemie.</li> <li>6. Wiadomość pojawia się w historii czatu.</li> </ol>

Tabela 4.10. Scenariusz PU-10 - Chat z użytkownikiem

<b>Przypadek użycia</b>	PU-10
<b>Nazwa przypadku</b>	Chat z użytkownikiem.
<b>Cel</b>	Chat z użytkownikiem.
<b>Aktorzy</b>	Rzemieślnik.
<b>Warunki wejściowe</b>	Inny użytkownik wykonał wcześniej PU-09 Chat z rzemieślnikiem lub rzemieślnik wykonuje PU-14 Obsługa zamówień.
<b>Warunki wyjściowe</b>	Rzemieślnik wysłał wiadomość do użytkownika.
<b>Scenariusze</b>	
<b>Przebieg główny:</b>	<ol style="list-style-type: none"> <li>1. Rzemieślnik naciska na ikonę z dymkiem.</li> <li>2. System przenosi rzemieślnika do ekranu chatu z wybranym użytkownikiem.</li> <li>3. Rzemieślnik wpisuje wiadomość.</li> <li>4. Rzemieślnik naciska ikonę strzałki.</li> <li>5. System zapisuje wiadomość w systemie.</li> <li>6. Wiadomość pojawia się w historii czatu.</li> </ol>

Bez możliwości dodawania ofert oraz usług przez rzemieślnika system nie spełniałby swojego przeznaczenia. Ta kluczowa funkcja przedstawiona jest poniżej w tabeli 4.11, a przypadki związane ogólnie z ofertami oraz usługami opisane są w tabelach 4.12, 4.13, 4.14, 4.16.

Tabela 4.11. Scenariusz PU-11 - Dodawanie ofert i usług

<b>Przypadek użycia</b>	PU-11
<b>Nazwa przypadku</b>	Dodawanie ofert i usług.
<b>Cel</b>	Dodanie usługi lub oferty.
<b>Aktorzy</b>	Rzemieślnik.
<b>Warunki wejściowe</b>	Brak.
<b>Warunki wyjściowe</b>	Rzemieślnik dodał ofertę lub usługę.
<b>Scenariusze</b>	
<b>Przebieg główny:</b>	<ol style="list-style-type: none"> <li>1. Rzemieślnik naciska na ikonę z plusem.</li> <li>2. System przenosi rzemieślnika do ekranu z dodaniem ofert lub usługi.</li> <li>3. Rzemieślnik wybiera pomiędzy usługą a ofertą.</li> <li>4. Rzemieślnik wprowadza dane oferty lub usługi.</li> <li>5. Rzemieślnik dodaje zdjęcie.</li> <li>6. Rzemieślnik naciska przycisk <i>Dodaj</i>.</li> <li>7. System sprawdza wprowadzone dane.</li> <li>8. System potwierdza poprawność danych.</li> <li>9. System zapisuje ofertę lub usługę w systemie.</li> <li>10. System informuje rzemieślnika o poprawnym dodaniu oferty lub usługi.</li> </ol>
<b>Przebieg alternatywny: <i>Dane nie spełniają wymagań.</i></b>	<p>8A-1. System informuje rzemieślnika o niepoprawnych danych.</p> <p>8A-2. Gość poprawia dane wskazane, przez system.</p> <p>8A-3. Powrót do punktu 6.</p>
<b>Przebieg alternatywny: <i>Rzemieślnik dodaje i zmienia rozmiar zdjęcia.</i></b>	<p>2A-1. Rzemieślnik naciska przycisk <i>Modyfikuj</i>.</p> <p>2A-2. Rzemieślnik modyfikuje zdjęcie.</p> <p>2A-3. Rzemieślnik naciska przycisk <i>Zapisz zmiany</i>.</p> <p>2A-4. Powrót do punktu 6.</p>

Tabela 4.12. Scenariusz PU-12 - Wyszukiwanie ofert i usług

<b>Przypadek użycia</b>	PU-12
<b>Nazwa przypadku</b>	Wyszukiwanie ofert i usług.
<b>Cel</b>	Znalezienie szukanej usługi oraz oferty.
<b>Aktorzy</b>	Użytkownik.
<b>Warunki wejściowe</b>	Brak.
<b>Warunki wyjściowe</b>	Użytkownik znalazł szukaną ofertę.
<b>Scenariusze</b>	
<b>Przebieg główny:</b>	<ol style="list-style-type: none"> <li>1. Użytkownik naciska na formularz wyszukiwania.</li> <li>2. Użytkownik wprowadza nazwę szukanej oferty lub usługi.</li> <li>3. Użytkownik naciska ikonę lupy lub ikonę lupy na klawiaturze.</li> <li>4. System wyszukuje oferty oraz usługi.</li> <li>5. System Wyświetla użytkownikowi znalezione oferty oraz usługi.</li> </ol>
<b>Przebieg alternatywny: Użytkownik konfiguruje wyszukiwanie.</b>	<p>2.5A-1. Użytkownik naciska na przycisk <i>filtruj</i>.</p> <p>2.5A-2. Użytkownik ustawia filtry wyszukiwania.</p> <p>2.5A-3. Użytkownik naciska przycisk <i>Akceptuj</i>.</p> <p>2.5A-4. Powrót do punktu 2.</p>
<b>Przebieg alternatywny: Brak wyników dla podanej frazy</b>	<p>5A-1. System informuje użytkownika o braku wyników.</p> <p>5A-2. Powrót do punktu 1.</p>

Tabela 4.13. Scenariusz PU-13 - Składanie zamówienia

<b>Przypadek użycia</b>	PU-13
<b>Nazwa przypadku</b>	Składanie zamówienia.
<b>Cel</b>	Złożenie zamówienia na ofertę lub usługę umieszczoną przez rzemieślnika.
<b>Aktorzy</b>	Użytkownik.
<b>Warunki wejściowe</b>	Użytkownik wykonuje PU-12 Wyszukiwanie ofert i usług lub PU-07 Przeglądanie profilu rzemieślnika.
<b>Warunki wyjściowe</b>	Użytkownik złożył zamówienie na ofertę lub usługę.
<b>Scenariusze</b>	
<b>Przebieg główny:</b>	<ol style="list-style-type: none"> <li>1. Użytkownik naciska przycisk <i>Złóż zamówienie</i>.</li> <li>2. Użytkownik uzupełnia dodatkowe informacje.</li> <li>3. Użytkownik naciska przycisk <i>Zamów</i>.</li> <li>4. System zapisuje zamówienie.</li> <li>5. System przenosi użytkownika do panelu użytkownika.</li> </ol>

Obsługa zamówienia przedstawiona jest w tabeli 4.14. Zamówienie musiało zostać wcześniej złożone przez użytkownika, co opisuje tabela 4.13. Rzemieślnik, jak i użytkownik, może wykonać kilka różnych akcji związanych z obsługą zamówienia. Rzemieślnik może zmienić statusu zamówienia na:

- zaakceptowane,
- odrzucone,
- w trakcie,
- gotowe do odbioru,
- wymaga doprecyzowania,
- odebrane.

Użytkownik może anulować zamówienie zmieniając jego status na:

- anulowane.

Dodatkowo, gdy zamówienie zmieni status na odebrane, użytkownik może wystawić ocenę z wykonanej usługi. Obydwaj aktorzy mogą dodać notatki oraz zdjęcia do zamówienia.

Tabela 4.14. Scenariusz PU-14 - Obsługa zamówień

<b>Przypadek użycia</b>	PU-14
<b>Nazwa przypadku</b>	Obsługa zamówień
<b>Cel</b>	Zarządzanie zamówieniami.
<b>Aktorzy</b>	Użytkownik i rzemieślnik
<b>Warunki wejściowe</b>	Użytkownik wcześniej wykonał PU-13 Składanie zamówienia
<b>Warunki wyjściowe</b>	Rzemieślnik obsłużył zamówienie.
<b>Scenariusze</b>	
<b>Przebieg główny:</b>	<ol style="list-style-type: none"> <li>1. Aktor otwiera kartę ze wszystkimi zamówieniami</li> <li>2. Aktor naciska kafelek ze szczegółami zamówienia</li> <li>3. Aktor obsługuje zamówienie</li> <li>4. Aktor naciska przycisk <i>Zapisz</i></li> <li>5. System zapisuje zmiany aktora</li> <li>6. System informuje aktora statusie wykonanych akcji</li> <li>7. System przenosi aktora do wszystkich zamówień</li> </ol>

Użytkownik w aplikacji, przy użyciu mapy może się zlokalizować oraz przejrzeć pobliskie warsztaty. Scenariusz opisujący tę funkcję znajduje się w tabeli 4.15.

Tabela 4.15. Scenariusz PU-15 - Lokalizacja oraz przeglądanie mapy

<b>Przypadek użycia</b>	PU-15
<b>Nazwa przypadku</b>	Lokalizacja oraz przeglądanie mapy
<b>Cel</b>	Lokalizacja oraz przeglądanie mapy
<b>Aktorzy</b>	Użytkownik
<b>Warunki wejściowe</b>	Użytkownik wyraził zgodę na lokalizację oraz ma włączony GPS
<b>Warunki wyjściowe</b>	Użytkownik się zlokalizował na mapie i poznał rzemieślników w otoczeniu
<b>Scenariusze</b>	
<b>Przebieg główny:</b>	<ol style="list-style-type: none"> <li>1. Rzemieślnik przechodzi do karty z mapą</li> <li>2. System wyświetla warsztaty rzemieślnicze</li> <li>3. Rzemieślnik naciska ikonę lokalizacji</li> <li>4. System lokalizuje użytkownika</li> <li>5. System wyświetla położenie użytkownika na mapie</li> <li>6. Użytkownik gestami przegląda mapę</li> </ol>

Tabela 4.16. Scenariusz PU-16 - Dodanie przedmiotu do obserwowanych

<b>Przypadek użycia</b>	PU-16
<b>Nazwa przypadku</b>	Dodanie przedmiotu do obserwowanych
<b>Cel</b>	Dodanie przedmiotu do obserwowanych
<b>Aktorzy</b>	Użytkownik
<b>Warunki wejściowe</b>	Użytkownik wykonał PU-06 Przeglądanie profilu użytkownika lub PU-12 Wyszukiwanie ofert i usług
<b>Warunki wyjściowe</b>	Użytkownik dodał przedmiot do obserwowanych
<b>Scenariusze</b>	
<b>Przebieg główny:</b>	<ol style="list-style-type: none"> <li>1. Rzemieślnik przechodzi do karty z mapą</li> <li>2. System wyświetla warsztaty rzemieślnicze</li> <li>3. Rzemieślnik naciska ikonę lokalizacji</li> <li>4. System lokalizuje użytkownika</li> <li>5. System wyświetla położenie użytkownika na mapie</li> <li>6. Użytkownik gestami przegląda mapę</li> </ol>

Dodawanie rzemieślnika do obserwowanych przebiega w sposób opisany w tabeli 4.17.

Tabela 4.17. Scenariusz PU-17 - Dodanie rzemieślnika do obserwowanych

<b>Przypadek użycia</b>	PU-16
<b>Nazwa przypadku</b>	Dodanie rzemieślnika do obserwowanych
<b>Cel</b>	Dodanie rzemieślnika do obserwowanych
<b>Aktorzy</b>	Użytkownik
<b>Warunki wejściowe</b>	Użytkownik wykonał PU-06 Przeglądanie profilu użytkownika lub PU-12 Wyszukiwanie ofert i usług
<b>Warunki wyjściowe</b>	Użytkownik dodał przedmiot do obserwowanych
<b>Scenariusze</b>	
<b>Przebieg główny:</b>	<ol style="list-style-type: none"> <li>1. Rzemieślnik przechodzi do karty z mapą</li> <li>2. System wyświetla warsztaty rzemieślnicze</li> <li>3. Rzemieślnik naciska ikonę lokalizacji</li> <li>4. System lokalizuje użytkownika</li> <li>5. System wyświetla położenie użytkownika na mapie</li> <li>6. Użytkownik gestami przegląda mapę</li> </ol>

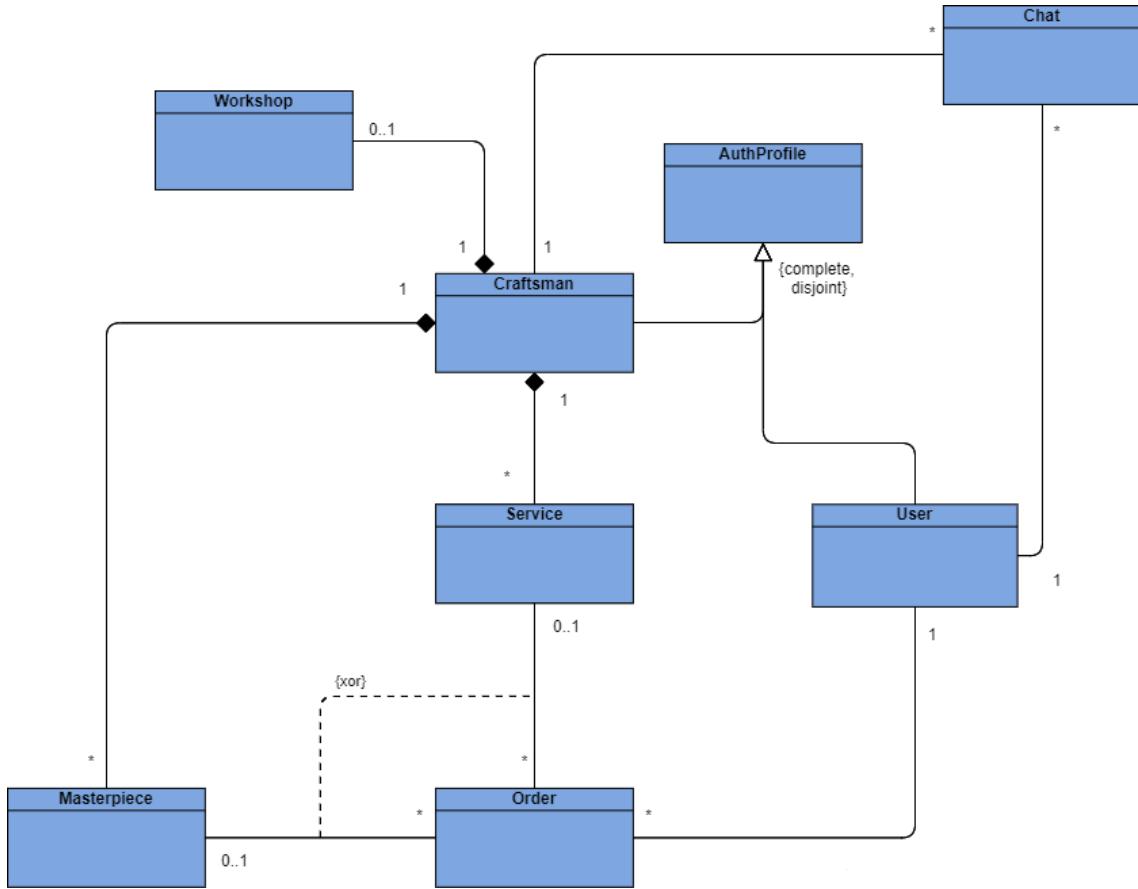
#### 4.5. Model domenowy

Na rysunku poniżej (rys. 4.2) przedstawiono model domenowy. Ukazuje on najważniejsze byty zawarte w aplikacji oraz to, w jaki sposób są ze sobą powiązane [26].

Klasy *Craftsman* oraz *User* są wyspecjalizowaniem klasy *AuthProfile*. Świadczy o tym dopisek przy strzałce dziedziczenia *{complete, disjoint}*, który oznacz, że klasa *AuthProfile* nie może istnieć samoistnie w systemie (*complete*) oraz że nie być specjalizowana jako *Craftsman* oraz *User* jednocześnie (*disjoint*).

Przechodząc do najważniejszego podmiotu systemu, komunikacji rzemieślnika z użytkownikiem, klasa ‘Chat’ odpowiedzialna jest za komunikator pomiędzy użytkownikiem oraz rzemieślnikiem. Dodatkowo dzięki wspomnianym powyżej ograniczeniom, konto rzemieślnika, założone na konkretny adres e-mail, nie może komunikować się z kontem użytkownika założonym na ten sam e-mail.

Kolejnym ważnym aspektem w aplikacji są zamówienia. Użytkownik, może złożyć zamówienie na serwis albo na przedmiot, czego reprezentacją są klasy *Order*, *Service*, *Masterpiece*, jednak zamówienie musi posiadać w sobie jedno z dwóch wspomnianych. Na diagramie gwarantuje to fragment języka *OCL*, języka zapisu ograniczeń w modelu obiektowym [59]. Zaznaczony jest on linią przerywaną na rys. 4.2 z dopiskiem *{xor}*.



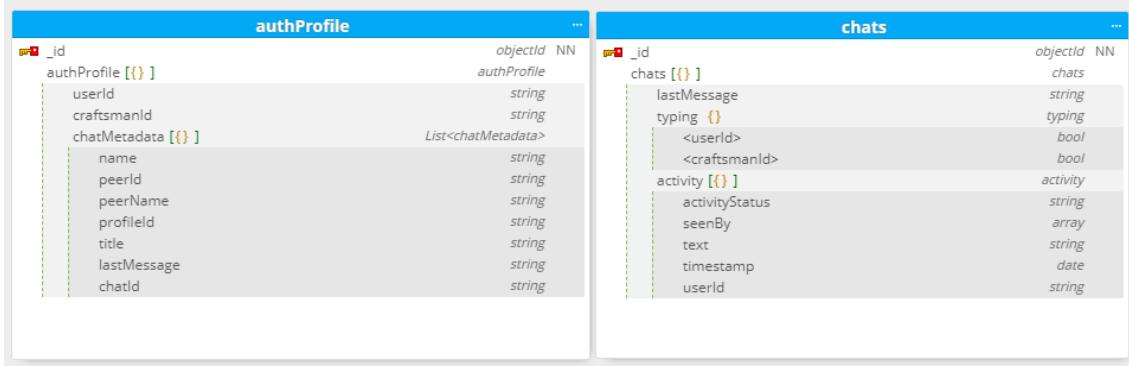
Rys. 4.2. Model domenowy aplikacji wspomagającej komunikację rzemieślnika z użytkownikami [źródło: opracowanie własne]

#### 4.6. Model fizyczny bazy danych

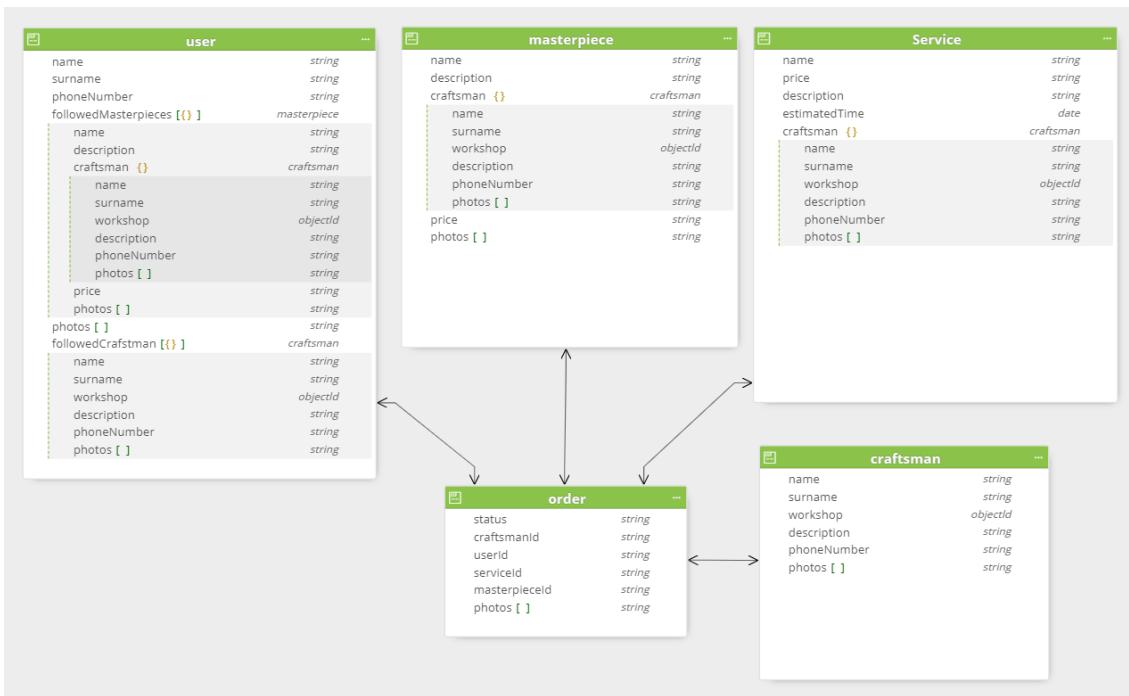
Z racji tego, że aplikacja oparta jest o dwa serwisy współpracujące ze sobą (Cloud Firestore oraz Flotiq), poniżej przedstawione zostaną dwa modele fizyczne baz danych. Pierwszy znajdujący się w systemie Cloud Firestore rys. 4.3, odpowiedzialny za przechowywanie wrażliwych informacji użytkowników oraz za komunikator pomiędzy nimi. Drugi znajdujący się we Flotiq rys. 4.4, przechowujący pozostałą część danych. Odwzorowanie dokładnej specyfikacji znajdujących się w *headless CMS* informacji, jest zadaniem niewykonalnym, z uwagi na fakt, braku dostępu do realnej struktury danych. Nazwanie owej specyfikacji relacyjną, również byłoby dużym nadużyciem, stąd też zdecydowano się przedstawić, tę strukturę tak samo jak strukturę bazy nierelacyjnej, gdyż w oczach autora jest ona jej najbliższa.

Poniżej rys. 4.3 przedstawia pola, które przechowuje Cloud Firestore. Lista dokumentów typu *chatMetadata* odpowiedzialna jest za przechowywanie informacji związanych z chatem. Dzięki temu wyświetlanie oraz poruszanie się pomiędzy nimi, nie wymaga dużej ilości kodu. Pola *UserId* oraz *CraftsmanId* odpowiedzialne są za przechowywanie identyfikatorów profili rzemieślnika oraz użytkownika, które znajdują

się we Flotiqu. W bazie nie ma potrzeby przechowywania takich informacji jak e-mail lub hasło użytkownika, gdyż obsługą tych danych zajmuje się Firebase Authentication. Są one bezpośrednio dostępne z poziomu warstwy prezentacji od razu po porwanym logowaniu. Po rejestracji w systemie specjalny identyfikator, *UID*, przydzielany jest profilowi, a następnie na jego podstawie, tworzony jest dokument *authProfile*, do którego przypisany jest identyfikator z Flotiga (*userId* albo *craftsmanId*).



Rys. 4.3. Model fizyczny bazy danych w Cloud Firestore [źródło: opracowanie własne]



Rys. 4.4. Model fizyczny bazy danych we Flotiqu [źródło: opracowanie własne]

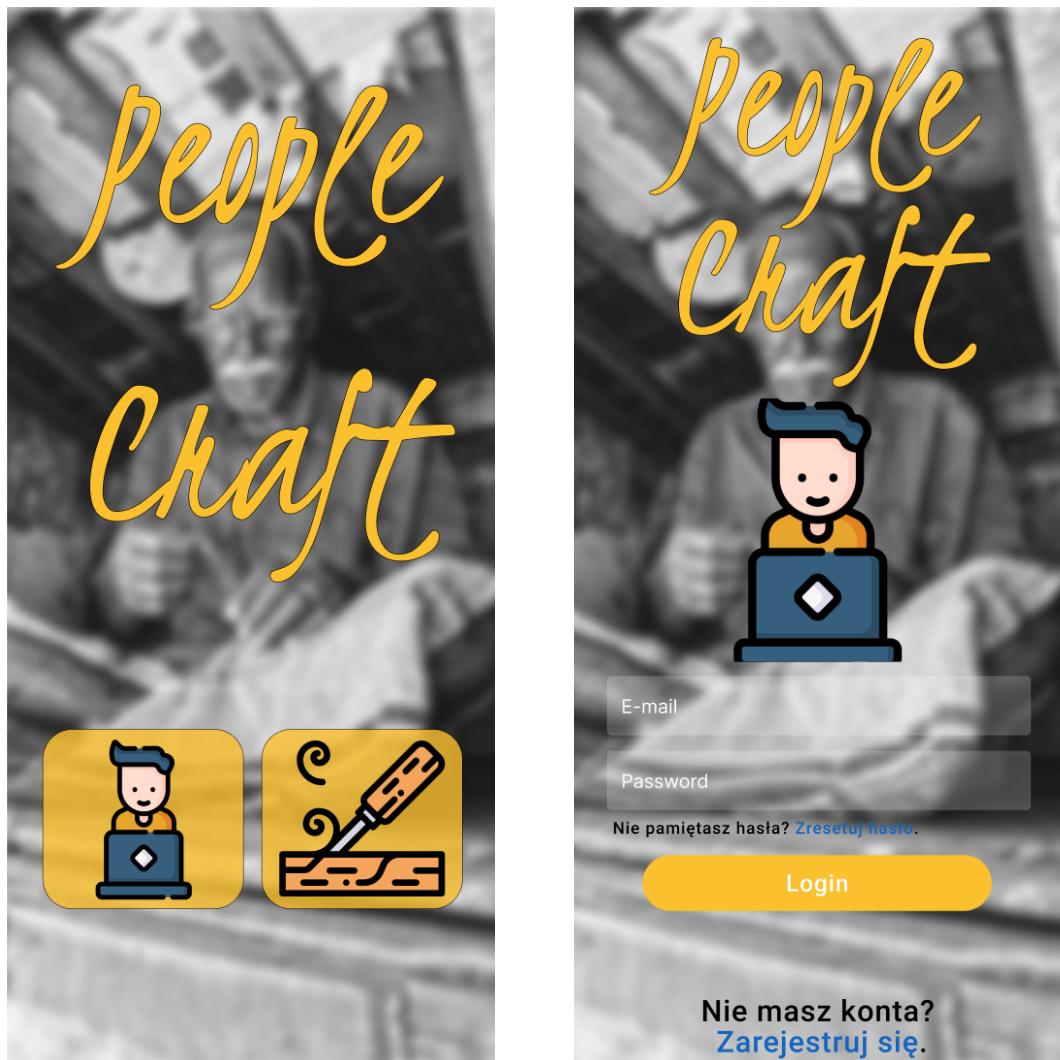
Na podstawie (rys. 4.2) zaprojektowano powyższy model fizyczny (rys. 4.4). Została w nim przedstawiona redundancja danych, jednak realnie w konkretnym CTD, znajduje się "link" do konkretnego obiektu w bazie. Zastosowano jednak takie uproszczenie, by lepiej przedstawić stan bazy danych we Flotiqu. Uproszczenie to zostało zastosowane z powodu możliwości punktów końcowych we Flotiqu. Parametr `?hydrate=1`, umieszczany w `queryParameters` (parametry zapytania), pozwala na pobranie z API, również zagnieżdżonych obiektów [13].

#### 4.7. Prototyp interfejsu użytkownika

Diagram przypadków użycia oraz model bazy danych były podstawą do stworzenia prototypów użytkownika. Podczas projektowania skupiono się na rozmieszczeniu elementów w aplikacji, jak i na zestawieniu kolorystycznym. W doborze tego drugiego pomogło narzędzie proponowane przez Google, Color Tool [31]. Oferowane jest ono w ramach Material Design [32], który jest przewodnim systemem projektowania zastosowanym w aplikacji. Zaprojektowana została zgodnie z wyznacznikami Material Design odnośnie dostępności [30].

W pracy przedstawiony zostanie przykładowy przebieg zaprojektowany w Figma. Prototypy zostały przygotowane dla iPhone 11 Pro Max. Przebieg przedstawił będzie logowanie, przegląd mapy oraz konfigurację profilu.

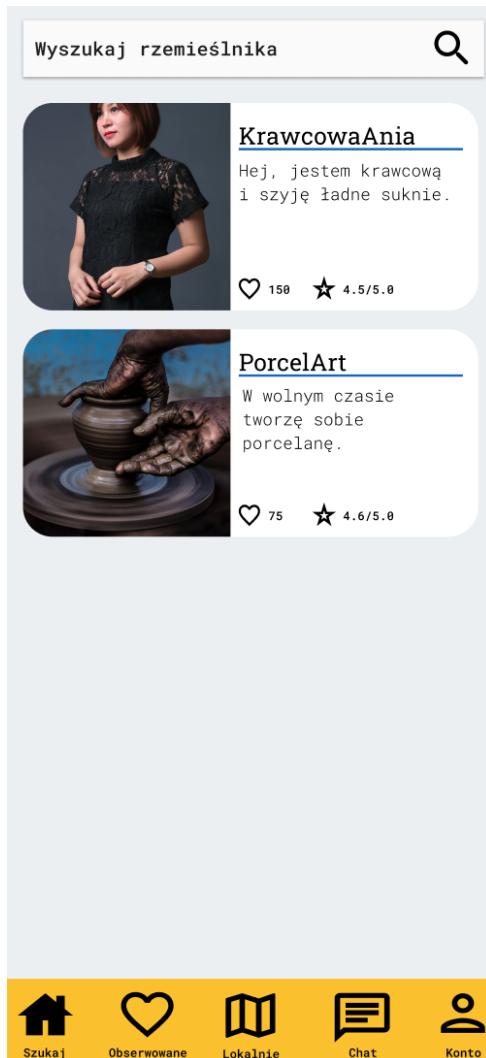
Użytkownik otwiera aplikację i ukazuje mu się ekran startowy (rys. 4.5). Następnie po naciśnięciu ikony użytkownika, przechodzi do ekranu logowania (rys.4.6).



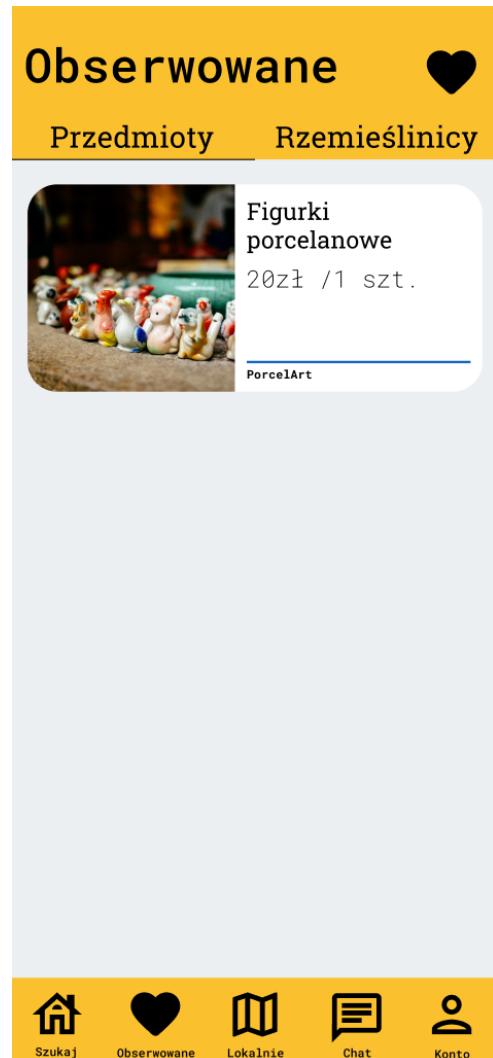
Rys. 4.5. Ekran startowy  
[źródło: opracowanie własne]

Rys. 4.6. Logowanie użytkownika  
[źródło: opracowanie własne]

Po poprawnym logowaniu, użytkownik przenoszony jest do ekranu wyszukiwania (rys. 4.7). Na dole został umieszczony pasek nawigacji, umożliwiający przechodzenie po poszczególnych funkcjach. Naciśnięcie ikony serca, przenosi użytkownika do obserwowanych (rys. 4.8).

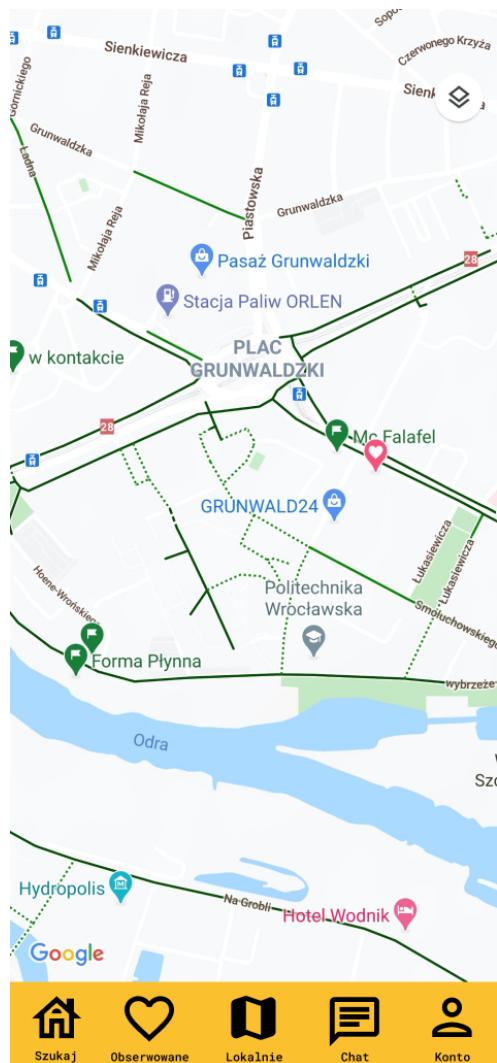


Rys. 4.7. Ekran wyszukiwania  
[źródło: opracowanie własne]

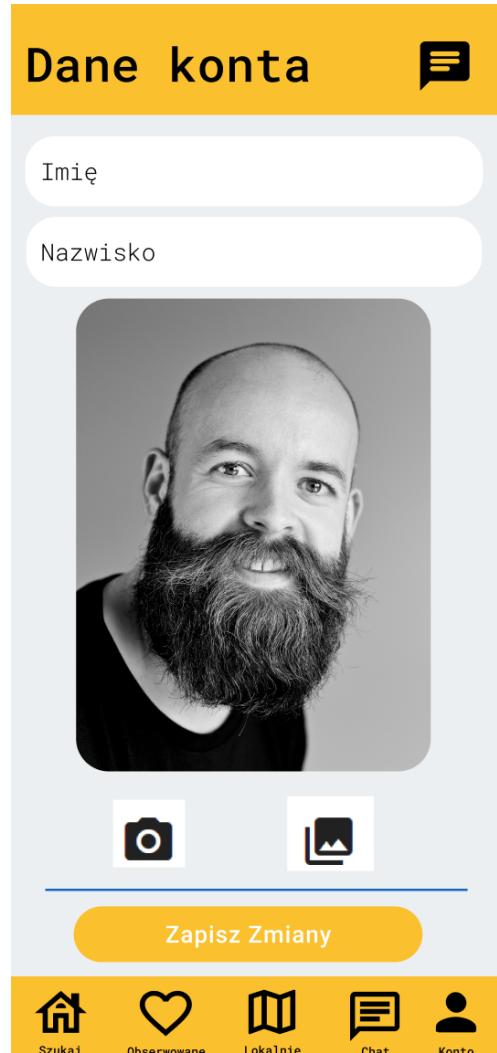


Rys. 4.8. Ekran obserwowanych  
[źródło: opracowanie własne]

Mapa pojawia się po naciśnięciu ikony mapy (rys. 4.9. Konfiguracja profilu (rys. 4.10) dostępna jest pod ikoną człowieka, a następnie po wybraniu z listy *Dane Konta*.



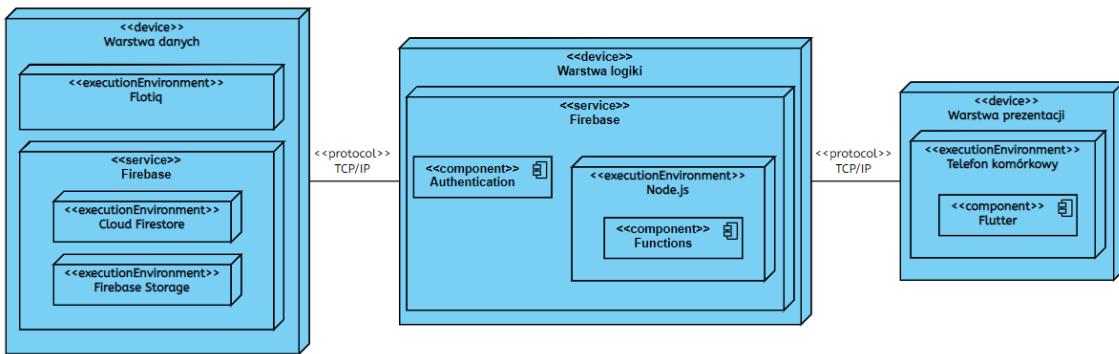
Rys. 4.9. Ekran mapy  
[źródło: opracowanie własne]



Rys. 4.10. Konfiguracja profilu  
[źródło: opracowanie własne]

#### 4.8. Diagram rozmieszczenia

Zgodnie z modelem architektury trójwarstwowej, aplikacja podzielono na interfejs użytkownika, przetwarzanie danych oraz składowanie danych. Mimo tego, że rozwiązanie typu *headless CMS* łączy w sobie warstwę logiki (przetwarzania danych), jak i warstwę danych (składowania danych) został umieszczony w warstwie danych, gdyż nie ma do niego dostępu bezpośrednio z warstwy prezentacji. Co odpowiada definicji warstwy składowania danych. Na poniższym diagramie rozmieszczenie (rys. 4.11) przedstawiono umiejscowienie poszczególnych modułów oraz połączenia między nimi.



Rys. 4.11. Diagram rozmieszczenia [źródło: opracowanie własne]

Za obsługę oraz uwierzytelnienie żądań wysyłanych z warstwy prezentacji odpowiedzialne są mikroserwisy (funkcje Firebase), znajdujące się w warstwie logiki. Klient w całości został zaimplementowany przy użyciu Fluttera. Wszystkie połączenia pomiędzy warstwami odbywają się za pomocą protokołu TCP/IP.

#### 4.9. Wzorce projektowe

W warstwie prezentacji zastosowano wzorzec wstrzykiwania zależności [27]. Wzorzec ten stosowany jest, by uniknąć silnych zależności, pomiędzy komponentami. Pozwala to na rozbicie dużych aktywności na mniejsze klasy, z których każda pełni określoną, atomową funkcję. Dzięki temu taka klasa może zostać użyta w wielu miejscach oraz łatwo ją używać wielokrotnie.

#### 4.10. Podsumowanie

Na bazie przedstawionego projektu zostanie zaimplementowany system wspomagający komunikację rzemieślnika z odbiorcą jego wyrobów. Wykonanie poprawnego projektu, jest bardzo istotnym krokiem w całym cyklu życia oprogramowania, gdyż pozwala dokładnie określić jego ramy, jak i pozwoli lepiej zrozumieć całemu zespołowi projektowemu, jaki cel należy osiągnąć.

## **5. Implementacja**

### **5.1. Warstwa Prezentacji**

**5.1.1. Najważniejsze rozwiązania**

**5.1.2. Napotkane problemy oraz sposób ich rozwiązania**

### **5.2. Warstwa Logiki**

**5.2.1. Najważniejsze rozwiązania**

**5.2.2. Napotkane problemy oraz sposób ich rozwiązania**

### **5.3. Testy**

**5.3.1. Scenariusze testowe**

**5.3.2. Wyniki testów**

### **5.4. Prezentacja Aplikacji**

### **5.5. Podsumowanie**

## **6. Zakończenie**

### **6.1. Zrealizowane prace**

### **6.2. Dalsze kroki rozwoju**

## Bibliografia

- [1] Amazon Web Services, *Amazon DynamoDB Fast and flexible NoSQL database service for any scale*, <https://aws.amazon.com/dynamodb/>. Ost. dost. 26 grudzień 2020.
- [2] Amazon Web Services, *Amzaon AWS*, <https://aws.amazon.com/>. Ost. dost. 12 grudzień 2020.
- [3] Amazon Web Services, *Aws amplify*, <https://aws.amazon.com/amplify/>. Ost. dost. 12 grudzień 2020.
- [4] Amazon Web Services, *AWS Cognito*, <https://aws.amazon.com/cognito/>. Ost. dost. 12 grudzień 2020.
- [5] Amazon Web Services, *AWS Lambda*, <https://aws.amazon.com/lambda/>. Ost. dost. 12 grudzień 2020.
- [6] Amazon Web Services, *Aws lambda pricing*, <https://aws.amazon.com/lambda/pricing/>. Ost. dost. 12 grudzień 2020.
- [7] Apple, *IOS*, <https://www.apple.com/pl/ios/ios-14/>. Ost. dost. 25 grudzień 2020.
- [8] Apple, *Objective-C*, <https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html>. Ost. dost. 25 grudzień 2020.
- [9] Apple, *Swift*, <https://swift.org/>. Ost. dost. 25 grudzień 2020.
- [10] Brendan Eich, *Javascript*, <https://developer.mozilla.org/pl/docs/Web/JavaScript>. Ost. dost. 25 grudzień 2020.
- [11] Cockburn, A., *Structuring use cases with goals*, Journal of Object-Oriented Programming. 1997, tom 10, 5, str. 56–62.
- [12] Codewave, *Flotiq*, <https://flotiq.com/>. Ost. dost. 26 grudzień 2020.
- [13] CodeWave, *Flotiq hydration*, <https://flotiq.com/docs/API/content-types/>. Ost. dost. 13 stycznia 2021.
- [14] Codewave, *Generating sdks*, <https://flotiq.com/docs/API/generate-package/>. Ost. dost. 26 grudzień 2020.
- [15] Codewave, *Search api in flotiq*, <https://flotiq.com/docs/Deep-Dives/search-api-deepdive/>. Ost. dost. 26 grudzień 2020.
- [16] Constantine, L.L., *Essential modeling: use cases for user interfaces. interactions*, Interactions. 1995, str. 34–46.
- [17] Drifty Co, *Ionic*, <https://ionicframework.com/>. Ost. dost. 12 grudzień 2020.
- [18] Drifty Co, *Ionic documentation*, <https://ionicframework.com/doc>. Ost. dost. 12 grudzień 2020.
- [19] Elasticsearch B.V., *Elasticsearch*, <https://www.elastic.co/>. Ost. dost. 26 grudzień 2020.

- [20] Facebook, *Core components and native components*, <https://reactnative.dev/docs/intro-react-native-components>. Ost. dost. 25 grudzień 2020.
- [21] Facebook, *React*, <https://pl.reactjs.org/>. Ost. dost. 25 grudzień 2020.
- [22] Facebook, *React Native*, <https://reactnative.dev/>. Ost. dost. 12 grudzień 2020.
- [23] Figma, *Don't sync to the cloud with adobe xd. work in the cloud with figma.*, <https://www.figma.com/figma-vs-adobe-xd/>. Ost. dost. 13 stycznia 2021.
- [24] Figma, *Figma*, <https://www.figma.com/>. Ost. dost. 13 stycznia 2021.
- [25] firebase.google.com, *Firebase core for flutter*, [https://pub.dev/packages/firebase\\_core](https://pub.dev/packages/firebase_core). Ost. dost. 26 grudzień 2020.
- [26] Fowler, M., *Patterns of Enterprise Application Architecture* (Addison-Wesley Longman Publishing Co., Inc., 2002).
- [27] Fowler, M., *Inversion of control containers and the dependency injection pattern*, <http://www.martinfowler.com/articles/injection.html>. 2008.
- [28] Ghost Fundation, *Ghost*, <https://ghost.org/>. Ost. dost. 26 grudzień 2020.
- [29] GitHub, *GitHub*, <https://github.com/>. Ost. dost. 13 stycznia 2021.
- [30] Google, *Accessibility*, <https://material.io/design/usability/accessibility.html#understanding-accessibility>. Ost. dost. 13 stycznia 2021.
- [31] Google, *color tool*.
- [32] Google, *Material DESIGN*, <https://material.io/>. Ost. dost. 13 stycznia 2021.
- [33] Google Developers, *Android*, [https://www.android.com/intl/pl\\_pl/](https://www.android.com/intl/pl_pl/). Ost. dost. 25 grudzień 2020.
- [34] Google Developers, *Android Studio*, <https://developer.android.com/studio>. Ost. dost. 13 stycznia 2021.
- [35] Google Developers, *Cloud firestore*, <https://firebase.google.com/docs/firestore>. Ost. dost. 26 grudzień 2020.
- [36] Google Developers, *Dart*, <https://dart.dev/>. Ost. dost. 12 grudzień 2020.
- [37] Google Developers, *Dart FAQ*, <https://dart.dev/faq>. Ost. dost. 25 grudzień 2020.
- [38] Google Developers, *Firebase*, <https://firebase.google.com/>. Ost. dost. 26 grudzień 2020.
- [39] Google Developers, *Firebase authentication simple, free multi-platform sign-in*, <https://firebase.google.com/products/auth>. Ost. dost. 26 grudzień 2020.
- [40] Google Developers, *Flutter*, <https://flutter.dev/>. Ost. dost. 25 grudzień 2020.
- [41] Google Developers, *Flutter FAQ*, <https://flutter.dev/docs/resources/faq>. Ost. dost. 12 grudzień 2020.
- [42] Google Developers, *Kotlin*, <https://developer.android.com/kotlin>. Ost. dost. 25 grudzień 2020.
- [43] Google Developers, *Maps sdk for android*, <https://developers.google.com/maps/documentation/android-sdk/overview>. Ost. dost. 23 grudzień 2020.
- [44] Google Developers, *Run apps on the android emulator*, <https://developer.android.com/studio/run/emulator>. Ost. dost. 13 stycznia 2021.

- [45] Google LCC, *Mapy*, <https://play.google.com/store/apps/details?id=com.google.android.apps.maps&gl=PL/>. Ost. dost. 12 grudzień 2020.
- [46] Hejlsberg, A., *TypeScript*, <https://www.typescriptlang.org/>. Ost. dost. 25 grudzień 2020.
- [47] inVerita, *Flutter vs native vs react-native: Examining performance*, <https://medium.com/swlh/flutter-vs-native-vs-react-native-examining-performance-31338f081980>. Ost. dost. 25 grudzień 2020.
- [48] James Gosling, *Java*, <https://www.java.com/pl/download/>. Ost. dost. 25 grudzień 2020.
- [49] JetBrains, *intelliJ idea ultimate*.
- [50] Larry Dignan, *Top cloud providers in 2020: Aws, microsoft azure, and google cloud, hybrid, saas players*, ZDnet. 2020. Ost. dost. 25 grudzień 2020.
- [51] Linus Torvalds, *Git*, <https://git-scm.com/>. Ost. dost. 13 stycznia 2021.
- [52] Miasto Jest Nasze, *Warszawscy rzemieślnicy*, <http://www.rzemieslnicy.waw.pl/>. Ost. dost. 12 grudzień 2020.
- [53] Microsoft, *Visual Studio Code*, <https://code.visualstudio.com/>. Ost. dost. 13 stycznia 2021.
- [54] Nader Dabit and Ashish Nanda, *Aws amplify release*, <https://aws.amazon.com/blogs/mobile/announcing-aws-amplify-flutter-developer-preview/>. Ost. dost. 12 grudzień 2020.
- [55] Netlify, *Headless cms a list of content management systems for jamstack sites*, <https://jamstack.org/headless-cms/>. Ost. dost. 26 grudzień 2020.
- [56] OLX Group, *Fixly dla wykonawców - zdobywaj zlecenia*, <https://play.google.com/store/apps/details?id=com.fixly.android.provider>. Ost. dost. 12 grudzień 2020.
- [57] OLX Group, *Fixly – do usług!*, <https://play.google.com/store/apps/details?id=com.fixly.android.user>. Ost. dost. 12 grudzień 2020.
- [58] OLX Group, *Olx - ogłoszenia lokalne*, <https://play.google.com/store/apps/details?id=pl.tablica>. Ost. dost. 12 grudzień 2020.
- [59] Place, F.N., *Object management group*, mars. 2000, tom 2005, str. 06–12.
- [60] Shamsee, N., Klebanov, D., Fayed, H., Afrose, A., Karakok, O., *CCNA Data Center DCICT 640-916 Official Cert Guide* (Pearson Education, 2015).
- [61] SMARTBEAR, *Openapi specification*, <https://swagger.io/specification/>. Ost. dost. 26 grudzień 2020.
- [62] StatCounter, *Mobile & tablet android version market share worldwide*, <https://gs.statcounter.com/android-version-market-share/mobile-tablet/worldwide/#monthly-201910-202010-bar>. Ost. dost. 25 grudzień 2020.
- [63] Strapi, *Strapi*, <https://strapi.io/>. Ost. dost. 26 grudzień 2020.
- [64] Strapi, *Strapi search*, <https://strapi.io/documentation/developer-docs/latest/content-api/parameters.html>. Ost. dost. 26 grudzień 2020.

- [65] Synopsys, *Compare repositories*, <https://www.openhub.net/repositories/compare>. Ost. dost. 13 stycznia 2021.
- [66] Wiredelta, *10 most popular prototyping tools of 2020*, <https://wiredelta.com/10-most-popular-prototyping-tools-of-2020/>. Ost. dost. 13 stycznia 2021.
- [67] Wohllebe, A., *Consumer acceptance of app push notifications: Systematic review on the influence of frequency*, <https://pdfs.semanticscholar.org/b754/71cdec4b3f219e62e0b13837103a7bfc9427.pdf>. 2020.

## Spis rysunków

2.1.	Wynik wyszukiwania w aplikacji Mapy [45]	7
2.2.	Przegląd dostępnych kategorii w aplikacji Fixly [57]	8
2.3.	Ekran wyszukiwania w aplikacji Fixly [57]	9
2.4.	Wygląd aplikacji Fixly dla wykonawców [56]	10
2.5.	Wygląd aplikacji OLX [58]	12
2.6.	Wygląd strony internetowej Warszawscy Rzemieślnicy [52]	13
3.1.	Wygląd aplikacji Fixly dla wykonawców [37]	18
3.2.	Panel administarcyjny Flotiq [12]	22
3.3.	Edytor Markdown w Ghost [28]	23
3.4.	Dodawanie <i>CTD</i> w strapi [63]	24
4.1.	Diagram przypadków użycia [źródło: opracowanie własne]	32
4.2.	Model domenowy aplikacji wspomagającej komunikację rzemieślnika z użytkownikiem [źródło: opracowanie własne]	47
4.3.	Model fizyczny bazy danych w Cloud Firestore [źródło: opracowanie własne]	48
4.4.	Model fizyczny bazy danych we Flotiqu [źródło: opracowanie własne]	48
4.5.	Ekran startowy	50
4.6.	Ekran startowy	50
4.7.	Ekran wyszukiwania	51
4.8.	Ekran obserwowanych	51
4.9.	Ekran mapy	52
4.10.	Konfiguracja profilu	52
4.11.	Diagram rozmieszczenia [źródło: opracowanie własne]	53

## **Spis tabel**

2.1.	Wady i zalety aplikacji Mapy . . . . .	13
2.2.	Wady i zalety aplikacji Fixly - Do usług! . . . . .	14
2.3.	Wady i zalety aplikacji Fixly dla wykonawców - zdobywaj zlecenia . . . . .	14
2.4.	Wady i zalety aplikacji OLX . . . . .	15
2.5.	Wady i zalety aplikacji Warszawscy Rzemieślnicy . . . . .	15
4.1.	Scenariusz PU-01 - Wylogowanie . . . . .	33
4.2.	Scenariusz PU-02 - Logowanie . . . . .	34
4.3.	Scenariusz PU-03 - Rejestracja . . . . .	35
4.4.	Scenariusz PU-04 - Konfiguracja profilu . . . . .	36
4.5.	Scenariusz PU-05 - Konfiguracja warsztatu rzemieślnika . . . . .	37
4.6.	Scenariusz PU-06 - Przegląd własnego profilu . . . . .	38
4.7.	Scenariusz PU-07 - Przeglądanie profilu rzemieślnika . . . . .	38
4.8.	Scenariusz PU-08 - Przeglądanie warsztatu rzemieślnika . . . . .	39
4.9.	Scenariusz PU-09 - Chat z rzemieślnikiem . . . . .	39
4.10.	Scenariusz PU-10 - Chat z użytkownikiem . . . . .	40
4.11.	Scenariusz PU-11 - Dodawanie ofert i usług . . . . .	41
4.12.	Scenariusz PU-12 - Wyszukiwanie ofert i usług . . . . .	42
4.13.	Scenariusz PU-13 - Składanie zamówienia . . . . .	43
4.14.	Scenariusz PU-14 - Obsługa zamówień . . . . .	44
4.15.	Scenariusz PU-15 - Lokalizacja oraz przeglądanie mapy . . . . .	45
4.16.	Scenariusz PU-16 - Dodanie przedmiotu do obserwowanych . . . . .	45
4.17.	Scenariusz PU-17 - Dodanie rzemieślnika do obserwowanych . . . . .	46