

**Politechnika Warszawska**

W Y D Z I A Ł M A T E M A T Y K I  
I N A U K I N F O R M A C Y J N Y C H



# Praca dyplomowa inżynierska

na kierunku Informatyka i Systemy Informacyjne

„Lost&Found” - wieloplatformowy i wielowarstwowy system dla rzeczy  
znalezionych

**Mateusz Szymański**

Numer albumu 305899

**Konrad Strzechowski**

Numer albumu 305891

**Michał Szadkowski**

Numer albumu 305894

promotor

Dr inż. Paweł Kotowski

WARSZAWA 2023



## Streszczenie

„Lost&Found” - wieloplatformowy i wielowarstwowy system dla rzeczy znalezionych

Niniejsza praca zawiera szczegółowy opis systemu Lost&Found, będącego serwisem ogłoszeniowym dla rzeczy znalezionych. Głównym celem systemu jest pomoc użytkownikom w odzyskiwaniu utraconych przedmiotów. Serwis gromadzi ogłoszenia na temat znalezionych lub zagubionych własności i pośredniczy w procesie ponownego łączenia ich z właścicielami. Użytkownicy mogą korzystać z systemu zarówno za pomocą aplikacji mobilnej jak i aplikacji webowej. Wszystkie funkcjonalności, dzięki którym potencjalny użytkownik może odzyskać utraconą własność, dostępne są w obu aplikacjach klienckich.

W poniższym dokumencie znajdziemy między innymi analizę biznesową, która pozwala poznać istniejące już rozwiązania wspólnie z ich zaletami i wadami. Praca zawiera również specyfikację, która wizualizuje i przedstawia szczegóły wymagań oraz funkcjonalności systemu Lost&Found. Dodatkowo autorzy umieścili w niej analizę potencjalnego ryzyka realizacji projektu.

Kolejne rozdziały pracy szczegółowo opisują architekturę systemu, w której wyróżnić możemy dwie główne warstwy. Warstwę składającą się z aplikacji klienckich oraz warstwę backendu o typowej strukturze mikroserwisów. Moduły tych warstw są kompleksowo przedstawione z uwzględnieniem ich zadań, zależności oraz zastosowanych do ich implementacji rozwiązań technologicznych. Realizowany projekt zawiera również szeroki pakiet testów jednostkowych, funkcjonalnych oraz integracyjnych. Proces ich implementacji oraz automatyzacji jest także zawarty i opisany w dokumencie.

Na końcu pracy zamieszczona została dokumentacja wdrożeniowa, instrukcja użytkowania oraz podsumowanie z realizowanego projektu. Dokumentacja wdrożeniowa opisuje kroki wymagane do skonfigurowania, zainstalowania i uruchomienia systemu. Instrukcja zawiera opis wszystkich funkcjonalności systemu z perspektywy użytkownika oraz instrukcje, jak korzystać z każdej z nich. Na zakończenie pracy autorzy pochylają się nad wnioskami związanymi z ukończeniem projektu oraz przedstawiają propozycje dalszego rozwoju systemu.

**Słowa kluczowe:** architektura mikroserwisów, serwis ogłoszeniowy, wieloplatformowość, geolokalizacja, React, React Native, ASP.NET Core, Docker, Azure



## Abstract

„Lost&Found” - cross-platform and n-tier system for found things

This thesis provides a comprehensive description of the Lost&Found system, which is a service for found things. The main goal of the system is to help users recover lost items. The service collects announcements about found or lost property and mediates the process of reuniting them with their owners. Users can access the system via both a mobile application and a web application. All the functionalities through which a potential user can recover lost property are available in both client applications.

Among other things, the following document includes a business analysis that allows you to learn about existing solutions together with their advantages and disadvantages. The paper also includes a specification that visualizes and presents the details of the requirements and functionality of the Lost&Found system. In addition, the authors have included an analysis of the potential risks of project implementation.

The next chapters of the work describe in detail the architecture of the system, in which we can distinguish two main layers. A layer consisting of client applications and a backend layer with a typical microservices structure. The modules of these layers are comprehensively presented including their tasks, dependencies and the technological solutions used for their implementation. The implemented project also includes an extensive package of unit, functional and integration tests. The process of their implementation and automation is also included and described in the paper.

The paper concludes with implementation documentation, a user manual and a summary from the completed project. The implementation documentation describes the steps required to configure, install and run the system. The manual contains a description of all the system's functionalities from the user's perspective and instructions on how to use each of them. At the end of the paper, the authors lean over the conclusions related to the completion of the project and provide suggestions for further development of the system.

**Keywords:** microservices architecture, advertising portal, cross-platform, geolocation, React, React Native, ASP.NET Core, Docker, Azure



# **Spis treści**

<b>1.</b>	<b>Wstęp</b>	<b>11</b>
1.1.	Podział pracy	12
1.2.	Cel systemu	12
1.3.	Analiza rynku	13
1.4.	Analiza ryzyka	14
<b>2.</b>	<b>Specyfikacja systemu</b>	<b>15</b>
2.1.	Wymagania funkcjonalne	15
2.2.	Wymagania niefunkcjonalne	18
<b>3.</b>	<b>Dokumentacja techniczna</b>	<b>19</b>
3.1.	Architektura systemu	19
3.2.	Aplikacje klienckie	21
3.2.1.	Aplikacja webowa	21
3.2.2.	Aplikacja mobilna	21
3.3.	Struktura mikroserwisów	22
3.3.1.	Moduł czatu	22
3.3.2.	Moduł autentykacji	23
3.3.3.	Moduł profilu	24
3.3.4.	Moduł publikacji	25
3.4.	Komunikacja	27
3.4.1.	Bezpieczeństwo i autoryzacja	27
3.4.2.	Dostępne punkty końcowe	28
3.5.	Zewnętrzne interfejsy	29
3.5.1.	Interfejs positionstack API	30
3.5.2.	Magazyn plików Azurite	30
<b>4.</b>	<b>Testy systemu</b>	<b>31</b>
4.1.	Testy jednostkowe	31
4.1.1.	Mikrowserwisy	32

4.1.2. Aplikacje klienckie . . . . .	33
4.2. Testy integracyjne . . . . .	33
4.2.1. Warstwa backendu . . . . .	33
4.2.2. Aplikacja webowa . . . . .	35
4.3. Testy E2E . . . . .	36
<b>5. Dokumentacja wdrożeniowa . . . . .</b>	<b>38</b>
5.1. Certyfikat SSL . . . . .	38
5.2. Serwer . . . . .	39
5.3. Konfiguracja . . . . .	39
5.3.1. Zapora sieciowa . . . . .	41
5.3.2. Nginx . . . . .	41
5.3.3. Uruchomienie systemu . . . . .	42
5.3.4. Budowanie aplikacji mobilnej . . . . .	44
<b>6. Instrukcja użytkowania . . . . .</b>	<b>45</b>
6.1. Wymagania sprzętowe . . . . .	45
6.2. Funkcjonalności z perspektywy użytkownika . . . . .	46
<b>7. Podsumowanie . . . . .</b>	<b>53</b>
7.1. Wnioski . . . . .	53
7.2. Perspektywy . . . . .	54

## 1. Wstęp

Utrata własności jest bolesnym doświadczeniem, dotyczącym w różnym stopniu każdego z nas. Powoduje ono zwiększyony stres, poczucie bezradności i bierności. Szczególnie trudne jest to dla osób, które utracą bardzo ważne dla nich przedmioty. Mogą to być dokumenty tożsamości, klucze lub inne cenne przedmioty sentymentalne.

Według danych zawartych na stronie miasta stołecznego Warszawy tylko 12% przedmiotów z powrotem trafia do ich właścicieli. W biurze rzeczy znalezionych na ulicy Dzielnej 15 prowadzone są szczegółowe statystyki dotyczące oddanych tam rzeczy znalezionych. W samym roku 2021 do placówki trafiło około 4400 dokumentów oraz 2200 przedmiotów innego rodzaju. Były to między innymi portfele, pieniądze, telefony, laptopy, aparaty fotograficzne, zegarki, tablety oraz rowery i biżuteria. Najważniejszą wiadomością jest jednak to, że to nie pojedyncze osoby przynoszą zguby do biura, a np. policjanci, personel lotniska, galerii handlowych, metra, czy też autobusu [4].

Gdy zwykły statystyczny obywatel znajdzie nienależący do niego przedmiot, z reguły decyduje się zwrócić go właścicielowi. Często podejmowanym działaniem jest dodanie ogłoszenia na Facebooku. Pozornie może wydawać się to dobrym miejscem ze względu, że według danych ponad 17 milionów Polaków korzysta z tego portalu [2]. Należy jednak podkreślić, iż nie jest on przystosowany do zarządzania publikacjami. Brakuje mu wielu funkcjonalności, które usprawniałyby proces wyszukiwania i odzyskiwania zagubionych własności. Dodatkowo na rynku nie istnieje wiele funkcjonalnych alternatyw mogących przejąć pulę użytkowników, korzystających obecnie z portali społecznościowych.

Wyciągając wnioski z powyższych danych i obserwacji, zauważalna jest potrzeba stworzenia aplikacji, która ułatwiłaby obywatelom proces odzyskiwania zgubionych przez nich przedmiotów. Takie aplikacje udostępnia system Lost&Found, który jest głównym tematem poniższej pracy dyplomowej. Lost&Found jest serwisem gromadzącym ogłoszenia, który wyróżnia się dzięki złożonym funkcjonalnościami filtrowania i wyszukiwania ogłoszeń. Serwis udostępnia użytkownikom aplikacje mobilną i webową.

## **1.1. Podział pracy**

System Lost&Found, w całości został zaprojektowany i zaimplementowany przez autorów poniższej pracy. Podczas całego procesu realizacji projektu podział pracy był równomiernie rozłożony wewnętrz z zespołu. Każdy miał swój udział w części praktycznej, teoretycznej oraz opisowej pracy. Poniżej sumarycznie przedstawiony został zakres obowiązków każdego z autorów pracy.

### **Mateusz Szymański**

- implementacja, testowanie oraz konfiguracja całej architektury warstwy backendu,
- automatyzacja procesu wytwarzania i testowania oprogramowania,
- opracowanie 3 i 4 rozdziału pracy dyplomowej.

### **Konrad Strzechowski**

- implementacja aplikacji mobilnej,
- konfiguracja i wdrożenie współdzielonej biblioteki pomiędzy aplikacjami klienckimi,
- opracowanie 1, 6 i 7 rozdziału pracy dyplomowej.

### **Michał Szadkowski**

- implementacja aplikacji webowej,
- wdrożenie systemu na serwer,
- opracowanie 2 i 5 rozdziału pracy dyplomowej.

## **1.2. Cel systemu**

Głównym celem opracowanego systemu Lost&Found jest pomoc osobom, które utraciły cenne dla nich przedmioty, poprzez połączenie ich z odpowiednimi znalazcami. Użytkownicy systemu posiadają możliwość dodawania ogłoszeń na temat znalezionych lub zagubionych własności. Specyfikując przy tym o jakim mieniu jest publikacja, wprowadzając opis, kategorie, zdjęcie i przybliżoną lokalizację w jakiej opisywany przedmiot został znaleziony lub utracony.

Dodatkowo system umożliwia dowolne przeglądanie ogłoszeń znalezionych i zaginionych przedmiotów z wybranej kategorii oraz w określonej okolicy. Jego wyróżniającą cechą jest możliwość

### **1.3. ANALIZA RYNKU**

stosowania złożonych filtrów i sortowań podczas wyszukiwania. Pozwala to użytkownikom na przeglądanie tylko tych ogłoszeń, które ich dotyczą. Ponadto osoby chcąc uzyskać więcej informacji na temat interesującego ich ogłoszenia, posiadają możliwość bezpiecznej komunikacji z autorem publikacji. Pozwala to znalazcy na bezpieczny proces weryfikacji, czy przedmiot należy do osoby określającej się jako jej właściciel.

Serwis dodatkowo zapewnia użytkownikom możliwość wpływania na promowanie ogłoszeń, udostępniając funkcjonalność w postaci polubień. Z systemu mogą korzystać dowolne osoby zamieszkające w Polsce, które posługują się językiem polskim przy użyciu aplikacji mobilnej lub webowej. Jednakże system został zaprojektowany i zaimplementowany w sposób umożliwiający łatwe rozszerzenie na użytkowników z innych państw.

#### **1.3. Analiza rynku**

Z obserwacji wynika, że na rynku obecnie nie istnieje popularny system z podobnymi funkcjonalnościami. Jak już zostało wspomniane, osoby, które utraciły jakiś przedmiot często zmuszone są do korzystania z różnego rodzaju grup na forach lub portalach społecznościowych, takich jak Facebook. Nie są to jednak miejsca służące odzyskiwaniu utraconych przedmiotów, zajmują się przeważnie szeroko pojętą komunikacją i rozrywką. Brakuje im odpowiednich narzędzi do filtrowania i sortowania ogłoszeń.

Wywieszanie fizycznych ogłoszeń w najbliższej okolicy, również nie jest rozwiązaniem skutecznym, ani tym bardziej wygodnym. Biura rzeczy znalezionych, pomimo swojej popularności posiadają znaczną wadę. W większości nie są funkcjonalne, brakuje im serwisu, gdzie publikowane byłyby wszystkie oddane tam przedmioty.

Konkludując nie istnieje jedno konkretne rozwiązanie zapewniające wygodę w odnajdywaniu zgubionych rzeczy, z którego użytkownicy mogliby korzystać niezależnie od lokalizacji. Takie rozwiązanie powinno być łatwo dostępne, przyjazne dla użytkownika i posiadać możliwości złożonego wyszukiwania ogłoszeń. System Lost&Found jest takim serwisem, dzięki czemu może wpaśować się w istniejącą lukę na rynku.

## 1.4. Analiza ryzyka

Potencjalne ryzyko realizacji projektu, zostało przeanalizowane podczas fazy analizy zgodnie z metodologią SWOT. Zagrożenia wewnętrzne i zewnętrzne zostały poddane szczegółowej ocenie prawdopodobieństwa ich wystąpienia. Wysokie prawdopodobieństwo oznaczane zostało symbolem  $W$ , średnie prawdopodobieństwo symbolem  $S$  i analogicznie niskie symbolem  $N$ . Wyniki przeprowadzonej analizy zostały przedstawione w tablicy 1.1.

Tabela 1.1: Wyniki analizy ryzyka przeprowadzonej metodologią SWOT.

	<b>Szanse</b>	<b>Zagrożenia</b>
<b>Wewnętrzne</b>	<ol style="list-style-type: none"> <li>1. Wszechstronność zespołu, możliwość adaptacji do dowolnie powierzonej roli.</li> <li>2. Zgrany i uzupełniający się zespół z wypracowanymi metodami pracy na bazie wcześniejszych wspólnych projektów i doświadczeń.</li> <li>3. Prostota aplikacji, system wykonuje konkretne zadanie.</li> </ol>	<ol style="list-style-type: none"> <li>1. Ograniczony zasób czasowy członków zespołu ze względu na inne projekty. <b>[W]</b></li> <li>2. Brak doświadczenia zespołu w tworzeniu aplikacji mobilnych. <b>[S]</b></li> <li>3. Prawdopodobieństwo, że wybrane technologie nie spełnią oczekiwania. <b>[N]</b></li> <li>4. Trudności w zastąpieniu jednej osoby z zespołu. <b>[N]</b></li> </ol>
<b>Zewnętrzne</b>	<ol style="list-style-type: none"> <li>1. Tworzony system dostępny zarówno jako portal internetowy i jako aplikacja na urządzenia mobilne - dopasowanie do preferencji użytkownika.</li> <li>2. Brak popularnego systemu o podobnych funkcjonalnościach na rynku.</li> <li>3. Wykorzystanie nowych technologii.</li> </ol>	<ol style="list-style-type: none"> <li>1. Ograniczone możliwości związane z promocją aplikacji - aplikacja potrzebuje większej liczby użytkowników, by lepiej spełniać swoje zadanie. <b>[S]</b></li> <li>2. Możliwość pojawienia się podobnego systemu w popularnej już aplikacji z dużą bazą użytkowników. <b>[N]</b></li> </ol>

## 2. Specyfikacja systemu

W celu określenia zakresu i sposobu realizacji systemu zebrano serię wymagań funkcjonalnych i niefunkcjonalnych dotyczących całości realizowanego projektu. Wymagania funkcjonalne definiują jakie funkcjonalności będą w systemie dostępne oraz jakie zadania może on realizować. Stanowią one podstawę do dalszego planowania i implementacji systemu [6]. Wymagania niefunkcjonalne określają jakość funkcjonalności dostarczanych przez system. Mogą one dotyczyć takich obszarów jak bezpieczeństwo, wydajność czy dostępność.

### 2.1. Wymagania funkcjonalne

Poniższa sekcja zawiera zebrane i pogrupowane tematycznie wymagania funkcjonalne systemu Lost&Found. Przedstawione są one w postaci tabel, w których kolumny opisują kolejno kto i jaką akcję może wykonać w systemie oraz jaki jest zakładany wynik wykonania tej akcji. W celu ustalenia priorytetu każdego z wymagań wykorzystano metodę MoSCoW, w której wyróżniamy 4 następujące kategorie, określające priorytet danego zadania: M – MUST (musi być), S – SHOULD (powinien być), C – COULD (może być), W – WON’T (nie będzie).

Tabela 2.1: Wymagania dotyczące przeglądania ogłoszeń.

Aktor	Akcja	Wynik
Użytkownik	przeglądać ogłoszenia z przedmiotami zgubionymi/znalezionymi [M]	wyświetlona zostaje lista przedmiotów znalezionych/zgubionych
	ustawić filtry wyszukiwania przedmiotów: data, lokalizacja, promień, nazwa, kategoria, stan ogłoszenia [M]	wyświetlona zostaje lista przedmiotów z ogłoszeniami spełniającymi określone kryteria
	ustawić sortowanie ogłoszeń według: daty, odległości [S]	wyświetlona lista zostaje posortowana według wybranego kryterium

Tabela 2.2: Wymagania dotyczące rejestracji, logowania i konta użytkownika.

Aktor	Akcja	Wynik
Użytkownik	założyć konto [M]	zostaje utworzone konto dla użytkownika z podanymi danymi
	zalogować się [M]	dane podane przez użytkownika są weryfikowane i jeżeli są poprawne to użytkownik uzyskuje dostęp do serwisu
	wyświetlić swój profil [M]	wyświetlony zostaje profil użytkownika, na który składają się: wyświetlana nazwa, imię i nazwisko, zdjęcie, opis, miasto
	zmienić hasło [W]	zmienione zostaje hasło wykorzystywane przez użytkownika do logowania
	edytować swój profil [M]	pokazany zostaje edytor, w którym użytkownik może zmienić dane na swoim profilu

Tabela 2.3: Wymagania dotyczące tworzenia i edycji ogłoszeń.

Aktor	Akcja	Wynik
Użytkownik	stworzyć ogłoszenie informujące o zaginięciu/znalezieniu przedmiotu [M]	wyświetlony zostaje kreator ogłoszenia pozwalający na podanie tytułu, opisu, zdjęcia, daty znalezienia/zgubienia, lokalizacji znalezienia/zgubienia i kategorii znalezionej lub zagubionej rzeczy
	wyświetlić wszystkie swoje ogłoszenia [S]	wyświetlone zostają wszystkie ogłoszenia utworzone przez użytkownika: najpierw aktywne a następnie ogłoszenia zamknięte posortowane według daty utworzenia ogłoszenia
	edytować swoje ogłoszenie [M]	wyświetlony zostaje edytor ogłoszenia analogiczny jak przy tworzeniu nowego ogłoszenia pozwalający na edycję szczegółów istniejącego ogłoszenia
	zamknąć swoje ogłoszenie [M]	ogłoszenie zmienia stan na zamknięte
	usunąć ogłoszenie [S]	usuwa ogłoszenie co skutkuje niewyświetlaniem ogłoszenia na listach

## 2.1. WYMAGANIA FUNKCJONALNE

Tabela 2.4: Wymagania dotyczące szczegółów ogłoszeń.

Aktor	Akcja	Wynik
Użytkownik	zobaczyć szczegółowy ogłoszenia [M]	wyświetlone zostają szczegółowe ogłoszenia takie jak: tytuł, opis, data znalezienia/zgubienia, data dodania ogłoszenia, kategoria, lokalizacja znalezienia /zgubienia przedmiotu, ocena ogłoszenia i stan
	ocenić ogłoszenie łapką w górę lub w dół [C]	ocena użytkownika zostaje dodana do puli ocen danego ogłoszenia

Tabela 2.5: Wymagania dotyczące profili innych użytkowników.

Aktor	Akcja	Wynik
Użytkownik	wyświetlić profil autora ogłoszenia [S]	pokazany zostaje profil użytkownika: wyświetlana nazwa, zdjęcie, opis, miasto i średnia ocen użytkowników
	ocenić użytkownika z poziomu jego profilu zostawiając ocenę w skali 0-5 i komentarz [C]	ocena zostaje dodana do puli ocen danego użytkownika i wyświetla się na profilu
	zobaczyć wszystkie oceny użytkownika [C]	wyświetlone zostają wszystkie oceny wraz z komentarzami zostawionymi na profilu użytkownika

Tabela 2.6: Wymagania dotyczące czatu.

Aktor	Akcja	Wynik
Użytkownik	z poziomu ogłoszenia rozpocząć czat w autorem ogłoszenia [S]	wyświetlony zostaje widok wszystkich czatów z aktywnym czatem z autorem ogłoszenia
	otworzyć okno z czatami [M]	wyświetlone zostaje okno ze wszystkimi wcześniejszymi czatami
	wybrać z okna czatów jeden z wcześniejszych czatów [M]	w oknie czatów jako aktywny zostaje ustawiony czat wybrany przez użytkownika
	wyświetlić poprzednie wiadomości w czacie [M]	po ustawieniu aktywnego czatu wyświetlane historia korespondencji pomiędzy użytkownikami
	napisać wiadomość [M]	nowa wiadomość zostaje dodana do czatu i wyświetlona zostaje drugiemu użytkownikowi

## 2.2. Wymagania niefunkcjonalne

W poniższej sekcji zawarte są wymagania niefunkcjonalne systemu Lost&Found. Dotyczą one obszarów innych niż funkcjonalności, tzn. użytkowanie, bezpieczeństwo, dostępność, wydajność i wsparcie.

Tabela 2.7: Lista wymagań niefunkcjonalnych systemu.

<b>Obszar wymagań</b>	<b>Nr.</b>	<b>Opis</b>
Użytkowanie	1.	Aplikacja poprawnie działa na przeglądarkach nie starszych wersji niż Firefox 90, Chrome 85, Safari 12.
	2.	Aplikacja jest dostępna do pobrania na telefon z systemem Android nie starszej wersji niż Android 10.
	3.	Aplikacja jest łatwa w użytkowaniu, design jest intuicyjny i przyjazny dla użytkownika.
	4.	Aplikacja jest dostępna w języku polskim.
Bezpieczeństwo	5.	System posiada serwis autoryzacyjny, zapewniający użytkownikom bezpieczny proces logowania.
	6.	Połączenie między użytkownikiem a serwerem jest szyfrowane, wykorzystuje protokół HTTPS.
Dostępność	7.	Aplikacja jest dostępna dla wszystkich osób na terenie Polski.
	8.	Aplikacja jest systemem dużej dostępności. Musi być ona dostępna przez 99% czasu. Przerwy techniczne nie mogą być dłuższe niż 2 godziny.
	9.	Aplikacja potrzebuje stałego dostępu do internetu.
Wydajność	10.	Aplikacja przetwarza zapytanie w nie dłużej niż 5 sekund.
	11.	Aplikacja może obsługiwać co najmniej 1000 aktywnych użytkowników jednocześnie.
Wsparcie	12.	Raz w tygodniu tworzona jest kopia zapasowa baz danych i przechowywana jest ona przez przynajmniej 1 miesiąc.

### **3. Dokumentacja techniczna**

Po określeniu niezbędnych funkcjonalności systemu, przystąpiliśmy do fazy projektowania mającej na celu transformacje wymagań na postać wykonywalną. Głównym celem było opracowanie szczegółowego planu dotyczącego budowy oraz implementacji systemu, który będzie spełniał wymagania i potrzeby użytkowników. Ponadto zależało nam na wypracowaniu modelu systemu, zapewniającego skalowalność, elastyczność oraz łatwość w utrzymaniu.

Niniejszy rozdział zawiera specyfikacje techniczną systemu Lost&Found, będącą efektem wspomnianych wyżej prac. Specyfikacja składa się z omówienia architektury systemu wraz z wyszczególnieniem jej warstw, modułów oraz głównych komponentów. Każdy moduł systemu został szczegółowo przedstawiony, uwzględniając przy tym opis jego zadań oraz wymagań technologicznych dotyczących implementacji. Dodatkowo omówione zostały sposoby i typy komunikacji. Dokumentacja techniczna uwzględnia również charakteryzacje zewnętrznych usług systemu wraz z opisami ich interfejsów.

#### **3.1. Architektura systemu**

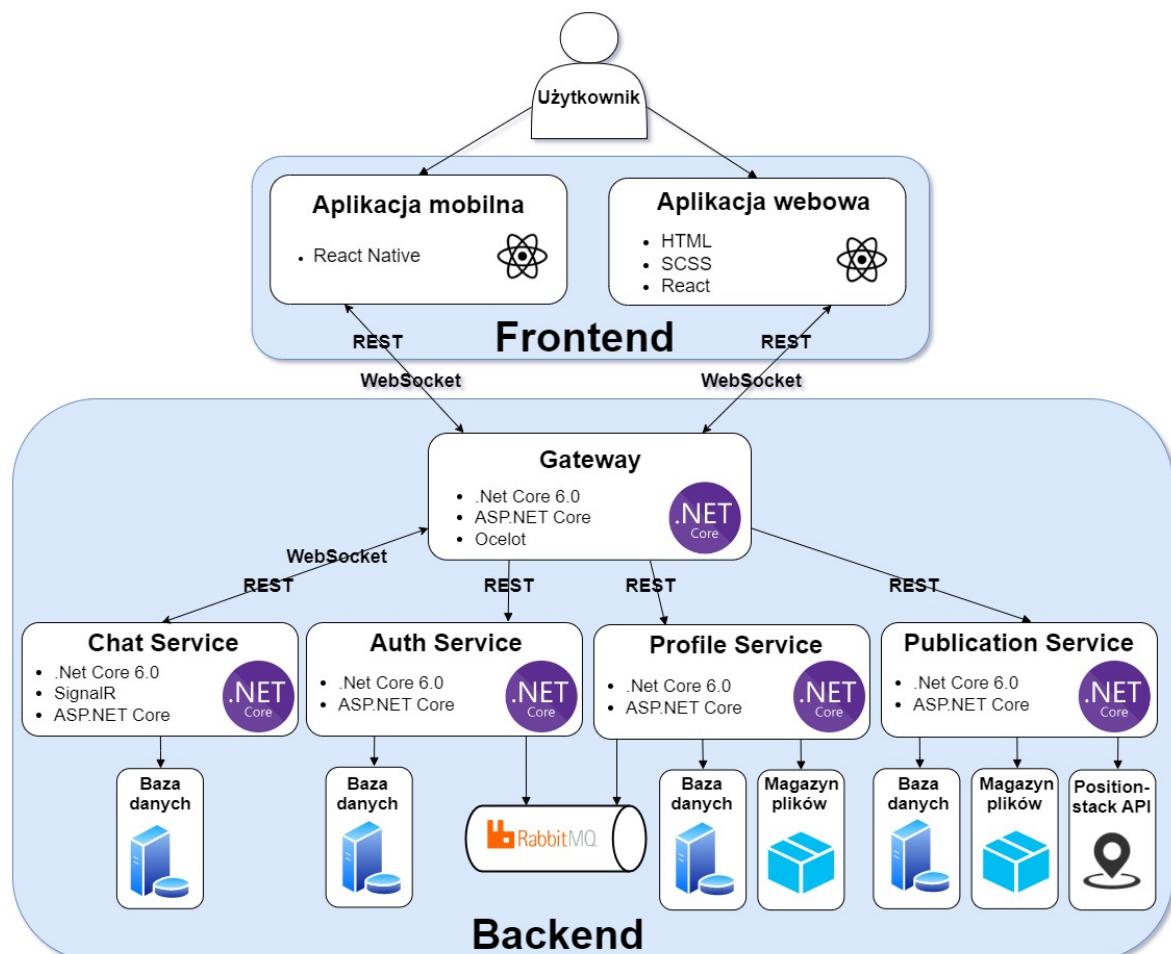
W architekturze systemu Lost&Found, wyróżniamy podział na dwie główne warstwy: frontendu i backendu. Poglądowy schemat architektury, zawierający wizualizacje podziału oraz poszczególne elementy warstw, został przedstawiony na rysunku 3.1.

Warstwa frontendu składa się z dwóch aplikacji klienckich: aplikacji webowej implementowanej przy użyciu biblioteki React.js oraz aplikacji mobilnej tworzonej z wykorzystaniem narzędzia React Native. Zadaniem aplikacji klienckich jest pobieranie danych od użytkowników oraz przekazywanie ich do warstwy backendu. Jednocześnie wyświetlając rezultaty zapytań użytkownikowi. Tylko przy użyciu tych modułów klienckich użytkownik posiada możliwość interakcji z systemem.

Architektura warstwy backendu ma postać typowej struktury mikroserwisów, tworzonych z wykorzystaniem platformy .NET 6.0. W jej skład wchodzą 4 niezależne moduły mikroserwisów odpowiadające za oddzielne funkcjonalności i korzystające z niepowiązanych zasobów. Są to kolejno:

- Moduł autentykacji (ang. Auth Service),
- Moduł czatu (ang. Chat Service),
- Moduł profilu (ang. Profile Service),
- Moduł publikacji (ang. Publication Service).

Wyszczególnione moduły nie komunikują się jednak bezpośrednio z aplikacjami klienckimi. Jak ilustruje rysunek 3.1, w procesie komunikacji uczestniczy dodatkowy moduł określany jako API Gateway (drzwi wejściowe). Jest to moduł będący pojedynczym punktem dostępu dla aplikacji klienckich. Jego głównym zadaniem jest przekierowywanie zapytań i wiadomości podczas procesu komunikacji pomiędzy aplikacjami klienckimi, a strukturą mikroserwisów. Dzięki zastosowaniu takiego rozwiązania, aplikacje klienckie nie są bezpośrednio zależne od modułów mikroserwisów i nie muszą nawet wiedzieć o ich istnieniu.



Rysunek 3.1: Poglądowy schemat architektury systemu Lost&Found

## 3.2. APLIKACJE KLIENCKIE

### 3.2. Aplikacje klienckie

Moduły z warstwy frontendu nazywamy aplikacjami klienckimi, które pozwalają użytkownikom na korzystanie z systemu Lost&Found. Klienci mają do dyspozycji aplikację webową oraz mobilną. Obie aplikacje są łatwe w obsłudze, a ich design jest intuicyjny i przyjazny dla użytkownika.

Aplikacje klienckie korzystają z biblioteki *commons*, w której zdefiniowane są zapytania potrzebne do komunikacji z warstwą backendu. Za jej pomocą aplikacje pobierają, edytują i zapisują wszystkie niezbędne dane.

#### 3.2.1. Aplikacja webowa

Moduł aplikacji webowej umożliwia użytkownikom korzystanie z systemu przy użyciu przeglądarki komputerowej. Aplikacja webowa oparta jest na bibliotece React.js. Pozwala ona na tworzenie interaktywnych interfejsów graficznych oraz łatwą wymianę danych pomiędzy komponentami [5]. Dodatkowo w procesie implementacji wykorzystywany jest język Typescript, który bazuje na języku Javascript i wzbogaca go o nowe funkcjonalności. Do obsługi strony typu SPA i przekierowywania zapytań, system wykorzystuje bibliotekę react-router. Jako bibliotekę dostarczającą klasy CSS wybrano Bootstrap w wersji 5.2. Natomiast język Sass służy do zmiany domyślnego stylu Bootstrap. Ponadto do obsługi dwukierunkowej komunikacji z modułem czatu, aplikacja korzysta z biblioteki microsoft/signalr w wersji 7.0.

#### 3.2.2. Aplikacja mobilna

Moduł udostępnia aplikację mobilną na urządzenia z systemem Android, która koresponduje z aplikacją webową, posiadając analogiczne funkcjonalności. Do jej wdrożenia i implementacji wykorzystywana jest platforma programistyczna React Native, która korzysta z biblioteki React.js [1]. Analogicznie do aplikacji webowej, w procesie implementacji wykorzystywany jest język Typescript.

Przy implementacji komponentów, takich jak nagłówki wykorzystywana jest biblioteka react-native-paper w wersji 5.1. Dodatkowo biblioteka react-native-snackbar w wersji 2.4 dostarcza funkcjonalności do komunikowania użytkownikowi błędów walidacji. Ikony występujące w aplikacji mobilnej pochodzą z biblioteki react-native-ionicons w wersji 4.6. Wrażliwe dane, takie jak tokeny autoryzacyjne, przechowywane są przy użyciu biblioteki react-native-encrypted-storage w wersji 4.0. Natomiast obsługa dwukierunkowych kanałów komunikacyjnych, tak samo jak w aplikacji webowej, oparta jest na bibliotece microsoft/signalr.

### 3.3. Struktura mikroserwisów

Moduły z warstwy backendu oparte są o platformę .NET 6.0 opracowywaną przez Microsoft. W procesie implementacji wykorzystywany jest język C# w wersji dziesiątej. Ponadto wspólną cechą wszystkich mikroserwisów jest biblioteka ASP.NET Core 6.0, stosowana do tworzenia internetowych aplikacji wykorzystujących protokół HTTP. Na schemacie architektury, umieszczonym na rysunku 3.1, zostały przedstawione moduły wraz z wykorzystywanymi przez nie zasobami. Można tam zauważyć, że każdy moduł ze struktury mikroserwisów posiada własną nierelacyjną bazę danych NoSQL. Konkretnie wykorzystują MongoDB jako nierelacyjny system do zarządzania bazą danych. MongoDB to dokumentowa baza danych, przechowująca dokumenty w formacie BSON (Binary JSON). Dodatkowo każdy element warstwy backendu, w celu zapewnienia skalowalności, jest skonfigurowany i skonteneryzowany przy użyciu platformy Docker. W dalszej części sekcji każdy moduł ze struktury mikroserwisów został opisany indywidualnie.

#### 3.3.1. Moduł czatu

Moduł czatu odpowiada za funkcjonalności związane z komunikacją pomiędzy użytkownikami. Pozwala uwierzytelnionym klientom wymieniać wiadomości w czasie rzeczywistym, a także przeglądać historię korespondencji. Do implementacji komunikacji w czasie rzeczywistym wykorzystuje on bibliotekę ASP.NET Core SignalR. Jest to biblioteka udostępniającą interfejs API do tworzenia zdalnych wywołań procedur serwer-klient (RPC) oraz używa zestawów WebSocket. Protokół WebSocket umożliwia tworzenie dwukierunkowych kanałów komunikacyjnych pomiędzy modułem czatu, a aplikacjami klienckimi.

Baza danych omawianego mikroserwisu składa się tylko z jednej kolekcji przechowującej dokumenty zawierające historię korespondencji pomiędzy poszczególnymi użytkownikami. Posiada ona złożony indeks, zdefiniowany przy użyciu unikalnych identyfikatorów użytkowników należących do czatu. Optymalizuje to najczęściej wykonywane operacje na kolekcji. Struktura dokumentu omawianej kolekcji została przedstawiona poniżej wraz z przykładowymi wartościami pól:

---

```
{
    "_id": 12,
    "creationTime": "2023-01-09T17:59:23.844+00:00",
    "exposedId": "3b2bafcd-b2fd-492b-b050-9b702765371",
    "members": [
        { "_id": "2b1bafcd-c2fd-482b-b050-9702765371" },
        { "_id": "4a1bafcd-b2fd-492b-b050-8702765371" }
    ],
}
```

### 3.3. STRUKTURA MIKROSERWISÓW

```
"containUnreadMessage": false,  
"messages": [  
    {  
        "content": "message 1",  
        "authorId": "2b1bafcd-c2fd-482b-b050-9702765371",  
        "creationTime": "2023-01-19T17:59:23.844+00:00",  
    }  
]
```

---

#### 3.3.2. Moduł autentykacji

Moduł autentykacji to mikroserwis odpowiedzialny za proces logowania i rejestracji użytkowników. W bezpieczny sposób przechowuje podstawowe dane, wymagane podczas procesu logowania. Wykorzystuje implementacje funkcji haszującej hasła bCrypt, a dokładniej implementacje zawartą w bibliotece BCrypt.Net-Next w wersji 4.0.0. Serwis odpowiada również za generowanie tokenów dostępu (ang. Access Token) i tokenów odświeżania (ang. Refresh Token) typu JWT (JSON Web Tokens). Wykorzystuje do tego systemowy pakiet platformy .NET System.IdentityModel.Tokens.Jwt. Tokeny te służą do uwierzytelniania w procesie komunikacji modułów klienckich z warstwą backendu. Podczas procesu rejestracji moduł publikuje wiadomości zawierające informacje o nowo zarejestrowanych użytkownikach do kolejki RabbitMQ.

Baza danych mikroserwisu autentykacji zawiera jedną kolekcję. Przechowywane są w niej dokumenty, zawierające dane użytkowników wymagane podczas procesu logowania. Kolekcja ta posiada pojedynczy indeks na polu zawierającym adres e-mail użytkownika, w celu zwiększenia wydajności najczęściej wykonywanych zapytań. Poniżej przedstawiona została, struktura dokumentu z omawianej kolekcji wraz z przykładowymi wartościami pól:

---

```
{  
    "_id": 1,  
    "creationTime": "2023-01-09T17:59:23.844+00:00",  
    "userId": "2b1bafcd-b2fd-492b-b050-9b702765371",  
    "username": "mateuszZ0pola12",  
    "email": "szymanski.mateusz0pole@gmail.com",  
    "passwordHash": "$2a$12$M.QuE0/bhaYWU6/i2VJUdJv33ZG",  
    "refreshToken": "bcJhbGciOiJIUzI1NiIsInR5cCLjHA"  
}
```

---

### 3.3.3. Moduł profilu

Moduł profilu użytkownika jest kolejnym mikroseriwsem występującym w systemie. Pozwala zarządzać profilami użytkowników oraz ich ocenami. Wykorzystuje usługę działającą w tle, która odbiera wiadomości, publikowane w kolejce RabbitMQ przez serwis autentykacji. Wiadomości te wykorzystywane są do tworzenia nowych profili użytkowników. Analogicznie do innych modułów posiada własną nierelacyjną bazę danych. Składa się ona z jednej kolekcji, w której przechowywane są dokumenty zawierające szczegóły profili użytkowników. Dokumenty te dodatkowo zawierają komentarze i oceny innych użytkowników. Kolekcja ta posiada indeks zdefiniowany przy użyciu identyfikatora użytkownika. Struktura dokumentu z tej kolekcji, została umieszczona poniżej:

---

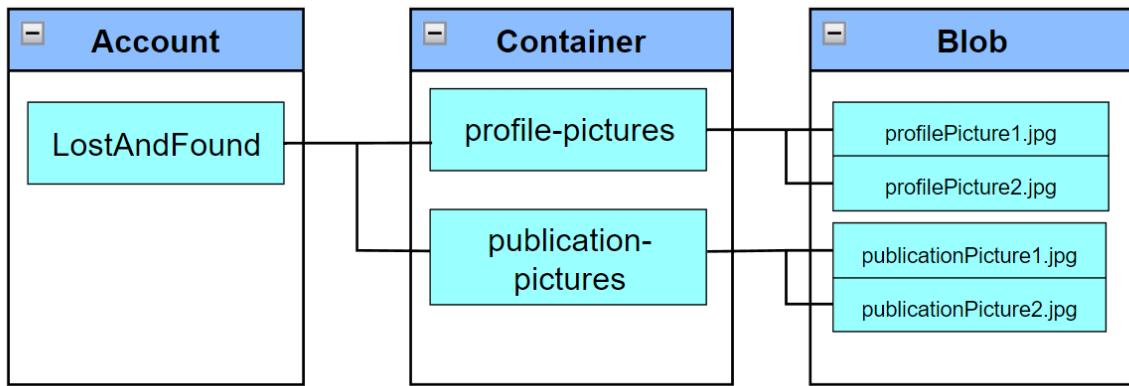
```
{
    "_id": 143,
    "creationTime": "2023-01-09T17:59:23.844+00:00",
    "userId": "2b1bafcd-b2fd-492b-b050-9b702765371",
    "username": "mateuszZOpola12",
    "email": "szymanski.mateuszOpole@gmail.com",
    "name": "Mateusz",
    "surname": "Szymanski",
    "description": "Lorem ipsum dolor sit amet, consectetur adipiscing elit.",
    "city": "Opole",
    "pictureUrl": "examplePicture1.png",
    "averageRating": 4,
    "comments": [
        {
            "content": "Dzieki za pomoc!",
            "rating": 4,
            "creationTime": "2023-02-09T17:59:23.844+00:00",
            "lastModificationDate": "2021-18-11T04:02:00Z",
            "authorId": "4c1bfcd-b2fd-492b-b0505371",
            "authorUsername": "testowyUser",
        }
    ]
}
```

---

Moduł profilu użytkownika oprócz bazy danych, wykorzystuje usługę Azurite do przechowywania zdjęć profilowych. Usługa ta jest zoptymalizowana pod kątem przechowywania ogromnych ilości niestrukturyzowanych danych. Na rysunku 3.2 przedstawiony został diagram zawierający konfiguracje zasobów magazynu obiektów blob dla systemu Lost&Found. Zawiera on konto

### 3.3. STRUKTURA MIKROSERWISÓW

Lost&Found wspólne dla modułu publikacji i modułu profilu. Zdjęcia profilowe użytkowników są rozróżniane za pomocą unikalnych nazw i są przechowywane jako pojedyncze obiekty blob w kontenerze *profile-pictures*. Aplikacja oparta o ASP.NET Core łączy się z magazynem plików przy użyciu biblioteki klienckiej środowiska .NET Azure.Storage.Blobs w wersji 12.14.1.



Rysunek 3.2: Konfiguracja zasobów magazynu obiektów systemu Lost&Found

#### 3.3.4. Moduł publikacji

Funkcjonalności związane z publikowaniem, modyfikowaniem i przeglądaniem ogłoszeń zawarte zostały w module publikacji. Komunikuje się on z zewnętrznym interfejsem positionstack API w celu przekształcania adresów na współrzędne geograficzne. Podobnie jak moduł profilu wykorzystuje usługę Azurite jako magazyn plików. Umieszczony na rysunku 3.2 kontener *publication-pictures*, wykorzystywany jest przez serwis publikacji do przechowywania zdjęć przedmiotów będących tematem ogłoszenia. Pozostałe funkcjonalności związane z magazynem plików są analogiczne, jak w przypadku mikroserwisu profilu.

Baza danych modułu składa się z dwóch kolekcji. Pierwsza z nich przechowuje dopuszczalne kategorie ogłoszeń. Jej przykładowa struktura została umieszczona poniżej:

```
{  
    "_id": 13,  
    "creationTime": "2023-01-07T19:01:27.829+00:00",  
    "exposedId": "Documents"  
    "displayName": "Dokumenty",  
}
```

### 3. DOKUMENTACJA TECHNICZNA

Druga kolekcja bazy danych gromadzi szczegółowe ogłoszenia umieszczone w serwisie. Dokumenty tej kolekcji zawierają między innymi opis, tytuł czy też typ zamieszczonej publikacji. Data oraz miejsce zdarzenia opisanego w ogłoszeniu również są w nim zawarte. Ponadto w dokumencie składowane są wszystkie oceny danej publikacji. Oceny te pochodzą od użytkowników systemu i mają wpływ na pozycjonowanie danego ogłoszenia w systemie.

Kolekcja ogłoszeń posiada złożony indeks zbudowany kolejno z pól przechowujących stan i typ ogłoszenia oraz szerokość i długość geograficzną miejsca zdarzenia. Są to pola dokumentów kolekcji, które są najczęściej używane w procesie filtrowania ogłoszeń. Poniżej przedstawiona została struktura dokumentu z kolekcji ogłoszeń wraz z przykładowymi wartościami pól:

---

```
{  
    "_id": 13,  
    "creationTime": "2023-01-09T20:39:23.279+00:00",  
    "exposedId": "2b1bafcd-c2fd-482b-b050-9702765371"  
    "title": "Zgubilem portfel, pomocy!",  
    "description": "Wczoraj wieczorem, zgubilem portfel w Warszawie na ulicy  
        Ludwika Waryńskiego. Proszę o pomoc, w portfelu znajdują się cenne dla  
        mnie dokumenty. Dla znalazcy czeka nagroda.",  
    "subjectPhotoUrl": "photourl1.png",  
    "incidentAddress": "Ludwika Waryńskiego 12, 00-655 Warszawa",  
    "incidentDate": "2022-01-09T20:39:23.279+00:00",  
    "subjectCategoryId": "Documents",  
    "subjectCategoryName": "Dokumenty",  
    "latitude": 52.17969,  
    "longitude": 20.97869,  
    "type": "LostSubject",  
    "state": "Open",  
    "aggregateRating": -1,  
    "lastModificationDate": "2023-01-09T20:39:23.279+00:00",  
    "votes": [  
        {  
            "voterId": "1a1bafcd-b2fd-492b-b050-9b70276371",  
            "rating": 1,  
            "creationDate": "2023-01-09T22:58:46.387+00:00",  
        }  
    ]  
}
```

---

### 3.4. Komunikacja

Możemy wyróżnić 3 typy komunikacji występujące w systemie Lost&Found. Pierwszym z nich jest komunikacja pomiędzy poszczególnymi mikroserwisami. Jest ona asynchroniczna i odbywa się przy wykorzystaniu brokera wiadomości RabbitMQ. Dzięki zastosowaniu tego typu komunikacji, serwisy mogą działać równolegle, co pozwala na lepsze wykorzystanie zasobów i szybsze działanie systemu. Jednakże największą zaletą tego rozwiązania jest uniknięcie zależności pomiędzy poszczególnymi mikroserwisami. Szczegóły wykorzystania brokera wiadomości zostały omówione przy opisie modułów profilu i autentykacji.

Pozostałe dwa typy komunikacji występują podczas transferu informacji pomiędzy warstwą backendu, a aplikacjami klienckimi. Transfer informacji odbywa się głównie przy wykorzystaniu protokołu połączeniowego HTTP (Hypertext Transfer Protocol). Jednakże w niewielkim stopniu wykorzystywany jest również protokół WebSocket, który umożliwia tworzenie dwukierunkowych kanałów komunikacyjnych pomiędzy modułem czatu, a aplikacjami klienckimi.

W dalszej części rozdziału opisano sposób zabezpieczenia istniejącej komunikacji pomiędzy warstwą frontendu i backendu. Dodatkowo dokładniej opisano architekturę REST (Representational State Transfer), która określa, jak komunikować się z serwerem przy użyciu protokołu HTTP.

#### 3.4.1. Bezpieczeństwo i autoryzacja

Proces uwierzytelniania w aplikacjach klienckich odbywa się przy wykorzystaniu tokenów JWT (JSON Web Token). Są one generowane przez moduł autentykacji i przekazywane do warstwy frontendu podczas procesu logowania. Tokeny są zaszyfrowane za pomocą klucza prywatnego i zawierają informacje o tożsamości użytkownika. Następnie przy każdym zapytaniu aplikacje klienckie dołączają ten token jako nagłówek *Authorization*. Umożliwia to poprawną identyfikację nadawcy i pozwala na przyznanie mu dostępu do odpowiednich zasobów. Token jest również wykorzystywany podczas tworzenia dwukierunkowych kanałów komunikacji opartych o protokół WebSocket.

W rzeczywistości podczas procesu logowania aplikacje klienckie otrzymują dwa tokeny: token dostępu (ang. access token) i token odświeżania (ang. refresh token). Pierwszy z nich wykorzystywany jest do identyfikacji użytkownika i został już omówiony. Drugi token pozwala uaktualnić i pobrać nowy token dostępu. Jest on wymagany w celu uniknięcia powtarzającego się procesu logowania, gdyż domyślnie ważność tokenu dostępu wynosi tylko 30minut. Ograniczenie czasowe jest zabezpieczeniem na wypadek przechwycenia tokenu przez osobę nieupoważnioną.

### 3.4.2. Dostępne punkty końcowe

Jak już zostało zaznaczone większość komunikacji w systemie odbywa się przy użyciu architektury REST. Poszczególne mikroserwisy udostępniają punkty końcowe (ang. endpoints), które są wykorzystywane przez moduły z warstwy frontendu przy użyciu API Gateway. Moduł ten jest pojedynczym punktem dostępu dla aplikacji klienckich i jego głównym zadaniem jest przekierowywanie zapytań do struktury mikroserwisów.

Poniżej wyszczególnione zostały punkty końcowe udostępniane przez warstwę backendu. Zostały one przedstawione wraz z krótkim opisem ich przeznaczenia. Ponadto tylko punkty dostępu przeznaczone do logowania i rejestracji nie wymagają uwierzytelniania. Natomiast wszystkie pozostałe zostały zabezpieczone przed nieautoryzowanym dostępem.

#### Punkty końcowe do zarządzania autoryzacją:

- **POST /account/register** - rejestracja nowego konta użytkownika.
- **POST /account/login** - logowanie użytkownika.
- **POST /account/refresh** - odświeżanie tokenu dostępu.
- **DELETE /account/logout** - wylogowanie użytkownika.
- **PUT /account/password** - zmiana hasła.

#### Punkty końcowe do zarządzania publikacjami:

- **GET /publication/categories** - pobranie dostępnych kategorii ogłoszeń.
- **PUT /publication** - pobranie listy publikacji.
- **POST /publication** - dodanie nowej publikacji.
- **GET /publication/{publicationId}** - pobranie szczegółów publikacji.
- **PUT /publication/{publicationId}** - edycja szczegółów publikacji.
- **PATCH /publication/{publicationId}** - zmiana stanu publikacji.
- **DELETE /publication/{publicationId}** - usunięcie publikacji.
- **PATCH /publication/{publicationId}/rating** - aktualizacja oceny publikacji.
- **PATCH /publication/{publicationId}/photo** - aktualizacja zdjęcia publikacji.
- **DELETE /publication/{publicationId}/photo** - usunięcie zdjęcia publikacji.

### 3.5. ZEWNĘTRZNE INTERFEJSY

#### Punkty końcowe do zarządzania profilem:

- **GET /profile** - pobranie danych własnego profilu.
- **PUT /profile** - edytowanie profilu.
- **PATCH /profile/picture** - aktualizacja zdjęcia profilowego.
- **DELETE /profile/picture** - usunięcie zdjęcia profilowego.
- **GET /profile/{userId}** - pobranie danych profilu innego użytkownika.
- **GET /profile/list** - pobranie podstawowych danych listy użytkowników.
- **GET /profile/{profileOwnerId}/comments** - pobranie komentarzy pod profilem.
- **POST /profile/{profileOwnerId}/comments** - dodanie komentarza.
- **PUT /profile/{profileOwnerId}/comments** - edycja komentarza.
- **DELETE /profile/{profileOwnerId}/comments** - usunięcie komentarza.

#### Punkty końcowe do zarządzania czatem:

- **GET /chat/** - pobranie historii korespondencji.
- **PATCH /chat/{chatMemberId}** - odczytanie odebranych wiadomości.
- **GET /chat/notification** - pobranie liczby nieodczytanych czatów.
- **POST /chat/message/{recipientId}** - wysłanie nowej wiadomości.
- **GET /chat/message/{recipientId}** - pobranie listy wiadomości danego czatu.

### 3.5. Zewnętrzne interfejsy

System wykorzystuje zewnętrzne usługi, do spełnienia niektórych z określonych wyżej wymagań funkcjonalnych. Między innymi korzysta z zewnętrznego interfejsu do przekształcania adresów na współrzędne geograficzne. Wykorzystuje również magazyny plików Azurite do przechowywania zdjęć profilowych i tych zawartych w ogłoszeniach. Poniżej indywidualnie przedstawione zostały poszczególne z nich.

### 3.5.1. Interfejs positionstack API

Moduł publikacji komunikuje się z zewnętrznym interfejsem API (Application Programming Interface) positionstack. Interfejs ten udostępnia proste i niezawodne narzędzia do dekodowania adresów statycznych na współrzędne geograficzne. Przykładowo pozwala na transformacje statycznego adresu „Komitetu Obrony Robotników 32, 02-148 Warszawa” na współrzędne geograficzne w postaci: szerokość geograficzna (ang. latitude): 52.17969 oraz długość geograficzna (ang. longitude): 20.97869. Pozwala to mikroserwisowi publikacji filtrować ogłoszenia z uwzględnieniem obszaru poszukiwań.

Dokładniej interfejs programistyczny aplikacji positionstack udostępnia punkt końcowy (ang. endpoint) GET akceptujący statyczny adres jako parametr zapytania. Ponadto interfejs positionstack jest zabezpieczony przed nieautoryzowanym użyciem. Wymaga ważnego klucza interfejsu API jako parametru każdego zapytania. Klucz ten ma postać niepowtarzalnego ciągu alfanumerycznego i jest powiązany z zarejestrowanym kontem w serwisie positionstack. Możliwe jest uzyskanie darmowego klucza do użytku niekomercyjnego pozwalającego na wysyłanie do 25000 zapytań w ciągu miesiąca.

### 3.5.2. Magazyn plików Azurite

Azurite to rozwiązanie firmy Microsoft służące do magazynowania obiektów. Jest zoptymalizowane pod kątem przechowywania olbrzymich ilości danych bez struktury. W systemie Lost&Found magazyn blob jest wykorzystywany do przechowywania zdjęć profilowych użytkowników oraz zdjęć zawartych w ogłoszeniach. W magazynie blob obiekty dostępne są za pośrednictwem interfejsu API REST usługi Azure Storage. System Lost&Found łączy się z magazynem plików przy użyciu biblioteki Azure.Storage.Blobs w wersji 12.14.0. Biblioteka ta udostępnia wygodny interfejs do wywoływanego zapytań usługi Azure Storage i jest dostępna na platformie .NET 6.0. Schemat konfiguracji magazynu Azurite został przedstawiony na rysunku 3.2.

Zdjęcia przechowywane w magazynie plików przesyłane są pomiędzy warstwami frontendu i backendu z wykorzystaniem protokołu HTTP. Do realizacji tej funkcjonalności wykorzystywane jest kodowanie Base64, które przekształca ciąg bajtów na ciąg znaków. Kodowanie Base64 zdefiniowane jest przez dokument RFC 4648 §4.

## **4. Testy systemu**

Podczas całego procesu tworzenia oprogramowania, wraz z powstawaniem kolejnych funkcjonalności w systemie, rozszerzał się również pakiet testowy systemu. Testy wywarzane były na każdym poziomie systemu. Zarówno w aplikacjach klienckich jak i w całej warstwie backendu. Ponadto testy systemu zostały zautomatyzowane przy użyciu usługi Azure, która umożliwia automatyzację procesów CI/CD (Continuous Integration/Continuous Deployment). W projekcie wykorzystywana jest do weryfikacji poprawności systemu przy każdej zmianie kodu źródłowego. Wprowadzanie zmian w danym module, skutkuje procesem automatycznego budowania i testowania jego funkcjonalności. Pozwala to na wychwycenie błędu natychmiast po jego pojawienniu się i szybką jego lokalizację. Rozdział w dalszej części przekrojowo opisuje wszystkie typy testów zawarte w systemie.

### **4.1. Testy jednostkowe**

Testowanie jednostkowe jest metodą wykorzystywaną do weryfikacji poprawności działania pojedynczych elementów (tzw. jednostek) systemu, takich jak funkcje, procedury czy też metody. Poszczególne fragmenty programu poddawane są testom, które je wykonują i sprawdzają czy ich przebieg i wynik jest zgodny z oczekiwany – pozytywnym lub negatywnym. Niepowodzenie działania kodu w określonych sytuacjach również może być oczekiwany wynikiem. Główną zaletą testów jednostkowych jest nieskomplikowany proces ich tworzenia oraz możliwość pełnej automatyzacji. Testy jednostkowe są również pewną formą dokumentacji systemu i pozwalają na lepsze zrozumienie kodu [7].

W naszym systemie testy jednostkowe zostały zaimplementowane zarówno w modułach warstwy backendu, jak i w aplikacjach klienckich. Dodatkowo, jak wspomniano wyżej, są one w pełni zautomatyzowane dzięki wykorzystaniu usługi Azure Pipelines.

#### 4.1.1. Mikrowserwisy

Każdy moduł z warstwy backendu posiada odpowiadający mu projekt testów jednostkowych. Testowanie jednostkowe odbywa się przy użyciu biblioteki xUnit, która jest domyślną biblioteką testową dla platformy .Net Core. Dodatkowo, przy implementacji testów, wykorzystywane były następujące biblioteki:

- Fluent Assertions,
- Moq.

Pierwsza z nich pozwala na tworzenie bardziej naturalnych zapisów asercji i walidacji. Natomiast biblioteka Moq jest przyjaznym i bardzo popularnym narzędziem, pozwalającym na zredukowanie zależności testowanej implementacji. Umożliwia stworzenie zaślepek (tzw. dublerów), które zastępują prawdzie obiekty naśladowując ich zachowanie.

Większość testów w systemie została stworzona z wykorzystaniem metody AAA (Arrange-Act-Assert). Metoda ta posiada wiele zalet i jest powszechnie stosowana. Dzieli test na trzy fazy: fazę przygotowania (ang. Arrange), fazę działania (ang. Act) oraz fazę assercji (ang. Assert) [3].

Poniżej przedstawiony został test jednostkowy zawarty w module serwisu autentykacji. Jego zadaniem jest sprawdzenie czy metoda generująca tokeny JWT zwraca je z odpowiednią datą przydatności (ang. expiration date). Test wykorzystuje wszystkie wspomniane wyżej biblioteki.

---

```
[Fact]
public void GenerateJwtToken_ReturnsTokenWithExpectedExpirationDate()
{
    // Arrange
    var utcDateNow = DateTime.UtcNow;
    var dateTimeProviderMock = new Mock<IDateTimeProvider>();
    mockedDateTimeProvider.Setup(m => m.UtcNow).Returns(utcDateNow);
    var jwtTokenGenerator = new JwtTokenGenerator(dateTimeProviderMock.Object);
    var expectedExpDate = utcDateNow.AddMinutes(30);

    // Act
    var jwtToken = jwtTokenGenerator.GenerateJwtToken(
        _secretForTests, "", "", expectedExpDate);
    var decToken = _jwtSecurityTokenHandler.ReadJwtToken(jwtToken);

    // Assert
    ((int)(decToken.ValidTo - expectedExpDate).TotalSeconds).Should().Be(0);
}
```

---

## 4.2. TESTY INTEGRACYJNE

### 4.1.2. Aplikacje klienckie

Do utworzenia testów wykorzystana została biblioteka React Testing Library dla aplikacji webowej oraz React Native Testing Library dla aplikacji mobilnej. Dostarczają one funkcjonalności pozwalające na symulacje testowanego elementu kodu oraz sprawdzenie poprawności. Zarówno aplikacja webowa jak i mobilna podzielone są na komponenty. Część komponentów odpowiedzialnych za wyświetlanie danych użytkownikowi w założony sposób (np. komponent renderowania profilu) posiada zestaw testów dotyczących poprawności wyświetlania danych. Na przykład w przypadku komponentu profilu sprawdzane jest czy wyświetlają się takie elementy jak: nazwa użytkownika, imię, nazwisko, miejscowości czy opis. Poniżej przedstawiony jest test aplikacji webowej. Tworzony jest komponent profilu i renderowany z wykorzystaniem przykładowych danych. Sprawdzane jest następnie czy wyświetlana jest nazwa użytkownika. Testy jednostkowe aplikacji mobilnej, ze względu na podobieństwa technologii, są bardzo zbliżone.

---

```
test("renders profile name", () => {
  var profile = new UserProfile();
  profile.name = "test name";
  render(<ProfileInner profile={profile} />, { wrapper: MemoryRouter });
  const name = screen.getByText(/test name/i);
  expect(name).toBeInTheDocument();
});
```

---

## 4.2. Testy integracyjne

Testy integracyjne mają za zadanie sprawdzanie poprawności interakcji między różnymi modułami lub komponentami aplikacji. W przeciwieństwie do testów jednostkowych testom integracyjnym poddawane są dwa lub więcej modułów. Podczas testowania integracyjnego zależy nam na sprawdzeniu czy komponenty poprawnie współpracują ze sobą. Przykładem może być współpraca między aplikacją ASP.NET Core, a bazą danych, ale nie tylko. Testom integracyjnym poddawana jest również kooperacja pomiędzy dwiema klasami lub metodami. Można więc stwierdzić, że testy integracyjne mają na celu wykrycie błędów podczas interakcji pomiędzy systemami lub jego częściami [7].

### 4.2.1. Warstwa backendu

Testy integracyjne pomiędzy modułami z warstwy backendu, implementowane są za pomocą platformy .Net Core. Podobnie jak w przypadku testów jednostkowych, każdy moduł z warstwy

backendu posiada odpowiadający mu projekt testów integracyjnych. Dodatkową analogią jest wykorzystanie bibliotek takich jak xUnit, oraz Fluent Assertions.

Na rysunku 3.1 przedstawiającym architekturę systemu zawarte zostały zasoby wykorzystywane przez poszczególne moduły. Przykładowo moduł publikacji wykorzystuje bazę danych MongoDB, kolejkę RabbitMQ i magazyn plików. Oznacza to, że do przeprowadzenia pełnych testów integracyjnych tego modułu, wymagane jest stworzenie testowych instancji tych zasobów. Zdecydowaliśmy się na wykorzystanie narzędzia Docker do tworzenia testowych instancji. Dokładniej logika uruchamiania i zarządzania kontenerami została obsłużona przy użyciu biblioteki Testcontainers.

Jest to rozwiązanie, które umożliwia uruchamianie kontenerów Docker w trakcie przeprowadzania testów integracyjnych. Dokładniej narzędzie dostarcza funkcjonalności pozwalającej przeprowadzać testy z jednorazowymi instancjami kontenerów Docker. Ponadto testcontainers zawiera dużą liczbę, wstępnie skonfigurowanych modułów, ale pozwala również na korzystanie z własnych obrazów. Zastosowanie tej biblioteki pozwala na równoległe wykonywanie testów integracyjnych oraz pozbycie się kłopotliwych zależności pomiędzy klasami testowymi.

Poniżej przedstawiony został sposób wykorzystania biblioteki testcontainers w testach integracyjnych systemu Lost&Found. Klasa IntegratioTestWebFactory w konstruktorze definiuje wykorzystywany kontener MongoDB, który następnie przed przeprowadzeniem testów z danej klasy testowej jest uruchamiany, a po zakończeniu testów zamknięty.

---

```

private readonly TestcontainerDatabase _container;

public IntegratioTestWebFactory()
{
    _container = new TestcontainersBuilder<MongoDbTestcontainer>()
        .WithDatabase(new MongoDbTestcontainerConfiguration
    {
        Database = "test_publication_db",
        Username = "mongo",
        Password = "mongo",
    })
    .Build();
}

public async Task InitializeAsync() => await _container.StartAsync();
public new async Task DisposeAsync() => await _container.DisposeAsync();

```

---

Następnie tak zdefiniowana klasa zawierająca proces uruchamiania kontenerów jest wyko-

## 4.2. TESTY INTEGRACYJNE

rzystywana przez poszczególne klasy testowe. Poniżej przedstawiony został jako przykład test integracyjny, zawarty w module publikacji. Jego zadaniem jest sprawdzenie czy proces pobierania kategorii z modułu publikacji zwraca poprawne dane. Ponadto test wykorzystuje wyżej zdefiniowaną klasę do obsługi kontenerów.

---

```
public class CategoriesControllerTests :  
    IClassFixture<IntegrationTestWebFactory<Program>>  
{  
    private readonly HttpClient _client;  
  
    public CategoriesControllerTests(IntegrationTestWebFactory<Program> factory)  
    {  

```

---

### 4.2.2. Aplikacja webowa

Wśród wszystkich komponentów aplikacji webowej możemy wydzielić takie, które w bezpośredni sposób prowadzą interakcję z komponentem dostarczającym kontekst użytkownika (dostarczającym informację o zalogowanym użytkowniku), jak np. pasek nawigacyjny czy komponenty odpowiedzialne za przenoszenie na inną stronę w przypadku innego kontekstu niż oczekiwany. Do utworzenia testów takich fragmentów potrzebne jest utworzenie komponentu kontekstu dostarczającego odpowiednie dane. Taki komponent musi występować wewnątrz komponentu routera odpowiedzialnego za poprawne działanie aplikacji typu SPA. Następnie sprawdzane jest jak testowany komponent działa w zależności od komponentu kontekstu.

W poniższym przykładzie tworzony jest komponent kontekstu użytkownika na podstawie sy-

mulowanych danych. Wykorzystując utworzony kontekst tworzony jest komponent paska nawigacyjnego, który w zależności od tego, czy użytkownik jest zalogowany, czy nie, ma się wyświetlać w różny sposób. Następnie sprawdzana jest poprawność wyświetlania komponentu poprzez wyszukanie w nim odpowiednich elementów.

---

```
test("renders logout button and profile link when logged in", () => {
  const user = new UsrCont();
  user.isLoggedIn = true;
  render(
    <userContext.Provider
      value={{ user, setUser: (arg: UsrCont) => {} }}
    >
      <Navbar />
    </userContext.Provider>,
    { wrapper: MemoryRouter }
  );
  var logoutBtn = screen.getByTestId("btnsignout");
  var profileLnk = screen.getByTestId("linktoprofile");
  var loginBtn = screen.queryByTestId("btnsignin");

  expect(logoutBtn).toBeInTheDocument();
  expect(profileLnk).toBeInTheDocument();
  expect(loginBtn).not.toBeInTheDocument();
});
```

---

### 4.3. Testy E2E

Testowanie E2E, nazywane również testowaniem końcowo-końcowym (ang. End-to-End testing), jest metodą testowania funkcjonalności poprzez symulowanie rzeczywistego użycia aplikacji. Ten rodzaj testowania sprawdza czy wszystkie składniki systemu działają razem zgodnie z oczekiwaniami, od interfejsu użytkownika po bazę danych. Pogadają na przeprowadzeniu symulacji działania systemu z poziomu użytkownika końcowego, przez przeprowadzenie operacji na interfejsie użytkownika. Testowanie wykonywane jest przy użyciu skryptów, które symulują interakcje użytkownika z aplikacją, takie jak klikanie przycisków czy wypełnianie formularzy. Ich celem jest wykrycie błędów lub problemów, które mogłyby nie zostać wykryte przez testy jednostkowe lub integracyjne oraz zapewnienie, że aplikacja działa w oczekiwany sposób [7].

W systemie Lost&Found testy E2E zostały zaimplementowane w aplikacji mobilnej z wyko-

### 4.3. TESTY E2E

rzystaniem narzędzia Detox. Pozwala ono na tworzenie międzyplatformowych testów na systemy iOS i Android. Symuluje działanie użytkownika poprzez wybieranie odpowiednich elementów, wpisywanie danych i wykonywanie kolejnych akcji.

Poniżej przedstawiony jest przykładowy test wykorzystywany w systemie Lost&Found. Detox uruchamia aplikację i weryfikuje czy wszystkie obiekty, takie jak przycisk do logowania, są widoczne na ekranie. Następnie wypełnia pola tekstowe podanymi danymi, aby zalogować się jako testowy użytkownik. Detox ponownie szuka spodziewanych elementów i w ten sposób weryfikuje czy logowanie powiodło się, a użytkownik został przeniesiony do następnego ekranu.

---

```
describe('Login', () => {
  beforeAll(async () => {
    await device.launchApp();
  });

  it('should have welcome screen', async () => {
    await expect(element(by.text('E-mail'))).toBeVisible();
    await expect(element(by.text('Haslo'))).toBeVisible();
    await expect(element(by.id('loginButton'))).toBeVisible();
    await expect(element(by.id('registerButton'))).toBeVisible();
  });

  it('login to app', async () => {
    const emailPlaceholder = element(by.id('emailPlaceholder'));
    await emailPlaceholder.typeText('test@gmail.com');
    await expect(emailPlaceholder).toHaveText('test@gmail.com');

    const passwordPlaceholder = element(by.id('passwordPlaceholder'));
    await passwordPlaceholder.typeText('12345678');
    await expect(passwordPlaceholder).toHaveText('12345678');

    const loginButton = element(by.id('loginButton'));
    await loginButton.tap();
  });

  it('should have posts screen', async () => {
    await expect(element(by.id('AddPostButton'))).toBeVisible();
  });
});
```

---

## 5. Dokumentacja wdrożeniowa

By dostarczyć system do potencjalnych użytkowników należało wybrać i zaimplementować jedno z wielu dostępnych na rynku rozwiązań umożliwiających udostępnienie aplikacji w internecie. Rozpatrywane rozwiązania obejmowały między innymi Azure App Service, Azure Virtual Machines i Google Compute Engine. Można je zaliczyć do dwóch grup: IaaS oraz PaaS. Ostatecznie wybrano Oracle Cloud Compute należącą do grupy IaaS, która dostarcza maszynę z pełnym dostępem administratora oraz przypisanym, publicznym adresem IP. Opcja ta wybrana została ze względu na wcześniejsze doświadczenie ze wdrażaniem podobnego systemu z wykorzystaniem tej usługi, ogólną prostotę tego typu rozwiązań oraz cenę. Posiada ona też swoje wady: konieczność konfiguracji większej liczby usług, np. zapory sieciowej czy konieczność samodzielnego wdrożenia SSL.

### 5.1. Certyfikat SSL

Użytkownicy w trakcie korzystania z systemu będą przesyłać do warstwy backendu dane potencjalnie poufne takie jak hasła, loginy, e-maile. Niezbędnym jest więc ich szyfrowanie. W tym celu wykorzystany został protokół HTTPS, który stanowi rozszerzenie protokołu HTTP o m. in. szyfrowanie za pomocą SSL. Wybrany dostawca usług chmurowych nie zapewnia serwera proxy, który szyfrowałby ruch i przekierowywał go do warstwy backendu. Potrzebne jest zatem samodzielne wygenerowanie certyfikatu SSL oraz wdrożenie szyfrowania na odpowiednich warstwach rozwiązania.

Jako organ do wygenerowania certyfikatu SSL wybrany został Let's Encrypt udostępniający swoje usługi w modelu non-profit. By wygenerować certyfikat powiązany z pewną domeną należy udowodnić posiadanie kontroli administratora nad maszyną, do której prowadzi ta domena. Taka metoda dobrze pasuje do stosowanego modelu wdrożenia. W tym celu wykorzystane zostało narzędzie Certbot dostarczane przez Let's Encrypt, które automatyzuje wszystkie kroki i generuje pliki zawierające certyfikat gotowy do wykorzystania.

Narzędzie Certbot pozawala na wygenerowanie certyfikatu powiązanego tylko z domeną, nie-

## 5.2. SERWER

możliwe jest powiązanie certyfikatu z adresem IP. Koniecznie jest zatem uzyskanie domeny, która przekierowywałaby na adres IP serwera, na której uruchomiony jest odpowiednia warstwa systemu. Domena pozwoli nie tylko na zwiększenie bezpieczeństwa, ale może również przyciągnąć potencjalnych użytkowników dzięki ułatwionemu udostępnianiu i zapamiętywaniu linku do serwisu. Do pozyskania domeny wykorzystano serwis FreeDNS umożliwiający proste i darmowe przypisanie subdomeny do adresu IP. Za jego pomocą przypisano subdomenę lostandfnd.my.to do publicznego adresu IP, czyli wcześniej utworzonego serwera w chmurze Oracle.

### 5.2. Serwer

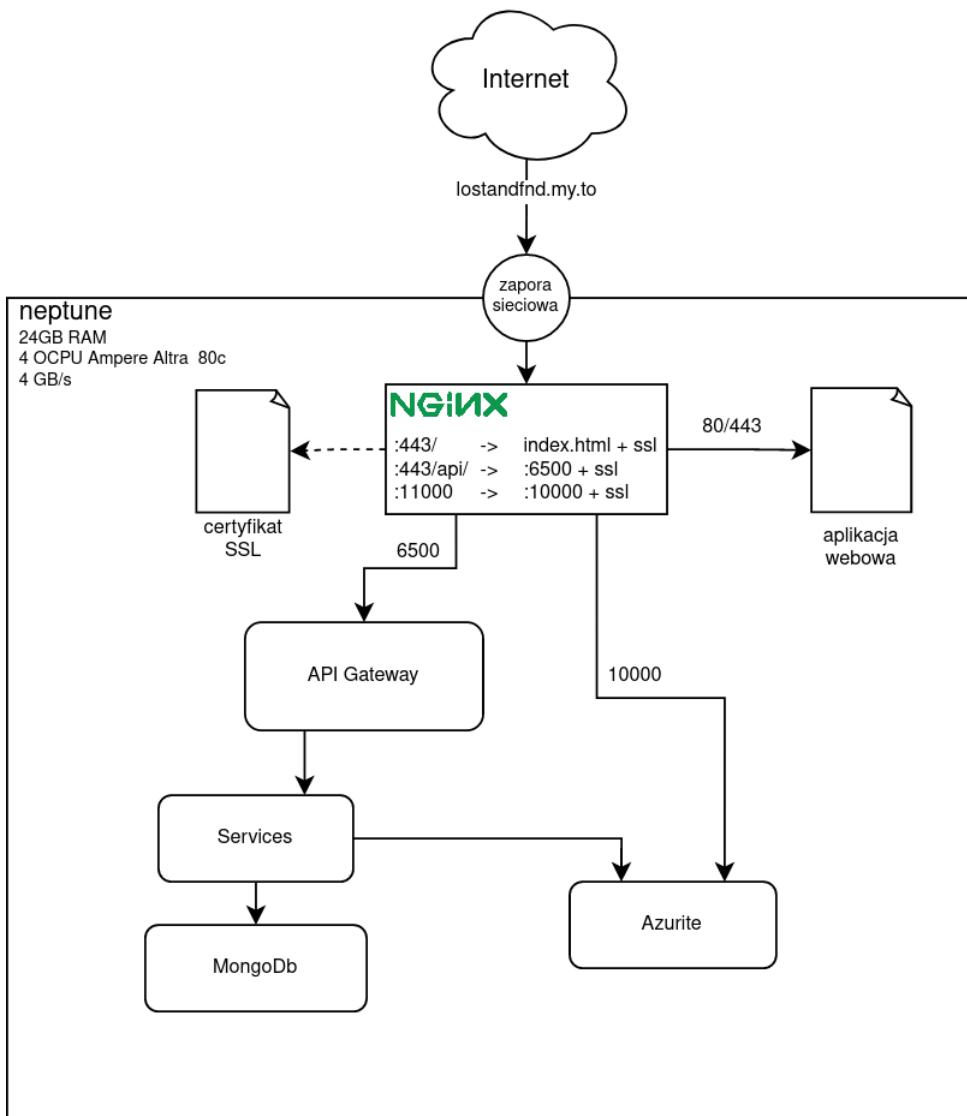
System wdrożony jest z wykorzystaniem pojedynczej instancji w ramach usługi Oracle Cloud Compute. Usługa ta na darmowym poziomie *Free Tier* pozwala na utworzenie maszyny w chmurze z następującą konfiguracją:

- pamięć RAM - 24GB,
- procesor - 4 rdzenie 3,0 GHz Ampere Altra 80C w architekturze ARM,
- przepustowość sieci - 4GB/s,
- pamięć dyskowa - 50GB.

Taka specyfikacja została wykorzystana do wdrożenia systemu. System operacyjny działający na maszynie to Ubuntu 22.04 LTS. Ponadto na serwerze uruchomione są wszystkie komponenty niezbędne do działania systemu.

### 5.3. Konfiguracja

Ze względu na wybrany model wdrożenia przed uruchomieniem systemu należy dokonać samodzielnej instalacji i konfiguracji innego oprogramowania wykorzystywanego do zarządzania ruchem sieciowym, udostępnienia aplikacji webowej oraz uruchomienia warstwy backendu. Poniższa sekcja zawiera opis wdrożenia na podstawie bieżącej konfiguracji, ale możliwe jest uruchomienie systemu w innych, nawet bardzo różniących się od zaprezentowanej, formach. Na rysunku 5.1 przedstawiony jest schemat konfiguracji.



Rysunek 5.1: Schemat konfiguracji wdrożenia

Przedstawiona konfiguracja opiera się przede wszystkim na dodatkowym module, wykorzystywanym tylko do wdrożenia, będącym punktem wejściowym całego systemu, który pełni rolę reverse proxy oraz dodatkowo szyfruje cały ruch wychodzący.

Do uruchomienia systemu w sugerowany sposób, poza niżej opisanymi krokami, niezbędne będzie zainstalowanie w systemie:

- Git,
- Node.js (w wersji 19.0),
- Docker.

## 5.3. KONFIGURACJA

### 5.3.1. Zapora sieciowa

Pierwszym wykorzystanym narzędziem jest zapora sieciowa, której zadaniem jest kontrolowanie ruchu. Stanowi ona najważniejszy komponent, jeśli chodzi o zapewnienie bezpieczeństwa rozwiązania backendu i zapobieganie innych nadużyć. Do kontrolowania konfiguracji zapory wykorzystano FirewallD domyślnie zainstalowany w systemie Ubuntu dostępnym w chmurze Oracle. Można jednak wykorzystać inne, dowolne narzędzie pozwalające na zablokowanie ruchu sieciowego na określonych portach. Konfiguracja zapory powinna wyglądać następująco:

- odblokowane są porty 80 (HTTP) i 443 (HTTPS),
- zablokowany jest port 6500 (API Gateway),
- zablokowane są porty wszystkich mikroserwisów,
- zablokowane są porty MongoDB oraz Azurite,
- pozostałe porty mogą być zablokowane lub odblokowane.

### 5.3.2. Nginx

Następnym narzędziem wykorzystywany do wdrożenia systemu jest Nginx. Jego rolą jest stanowienie, wcześniej zapowiedzianego, punktu wejściowego do systemu. Dzięki szerokim możliwościom będzie on działał nie tylko jako reverse proxy i szyfrował ruch, ale ponadto działał jako serwer dla aplikacji webowej. Dokładniej cały ruch przechodzący przez zaporę dociera tylko do modułu wejściowego, a ten przekierowuje go do odpowiednich modułów lub zwraca odpowiednie pliki. Odpowiada również za szyfrowanie ruchu wychodzącego do i przychodzącego z Internetu za pomocą wcześniej wygenerowanego certyfikatu SSL. Zastosowanie jednego, osobnego modułu odpowiedzialnego za te zadania pozwala na odciążenie modułu API Gateway z konieczności szyfrowania i ułatwia ewentualne skalowanie poziomu całego rozwiązania. Konfiguracja Nginxa jest następująca:

- cały ruch na porcie 443 jest szyfrowany za pomocą certyfikatu SSL,
- ruch na porcie 443 pod ścieżką /api/ jest przekierowywany do modułu API Gateway na porcie 6500,
- ruch na porcie 443 pod ścieżką / jest przekierowywany do aplikacji webowej,
- ruch na porcie 11000 jest szyfrowany i przekierowywany do Azurite na porcie 10000,
- ruch na porcie 80 pod ścieżką /apk/ jest przekierowywany do pobrania aplikacji mobilnej.

Poniżej przedstawiony jest najważniejszy fragment pliku konfiguracyjnego, dotyczący portu 443, przez który odbywa się największa część ruchu.

---

```

server {

    listen          443 ssl;
    listen          [::]:443 ssl;
    server_name     lostandfnd.my.to www.lostandfnd.my.to;

    ssl_certificate /etc/letsencrypt/live/lostandfnd.my.to/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/lostandfnd.my.to/privkey.pem;
    ssl_protocols    TLSv1 TLSv1.1 TLSv1.2;
    ssl_ciphers      HIGH:!aNULL:!MD5;

    root /home/ubuntu/lostandfnd-deploy/front/;
    access_log /home/ubuntu/lostandfnd-deploy/front/haccess.log;
    error_log /home/ubuntu/lostandfnd-deploy/front/ngerror.log;

    location /api/ {
        proxy_set_header Host $Host;
        proxy_http_version 1.1;
        proxy_set_header Connection $http_connection;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header x-forwarded-for $remote_addr;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_pass http://127.0.0.1:6500/;
        proxy_redirect http://127.0.0.1:6500/ https://$server_name/;
    }
    location / {
        try_files $uri /index.html;
    }
}

```

---

Poza opisanymi funkcjami moduł ten służy do monitorowania ruchu. Każde zapytanie na porcie 443 zapisywane jest w pliku haccess.log wraz z adresem IP źródła zapytania, datą oraz danymi przeglądarki. Pozwoli to na eliminowanie niechcianego, potencjalnie szkodliwego ruchu.

### 5.3.3. Uruchomienie systemu

Po poprawnie wykonanej konfiguracji opisanej powyżej można przystąpić do wdrożenia zasadniczej części systemu, na który składają się aplikacja webowa, aplikacja mobilna oraz warstwa backendu. Wdrożenie aplikacji mobilnej ogranicza się do udostępnienia pliku .apk umożliwiają-

### 5.3. KONFIGURACJA

cego zainstalowanie aplikacji na telefonie, co zostało opisane w poprzednim rozdziale.

Wdrożenie aplikacji webowej polega na pobraniu lub aktualizacji niezbędnych modułów, komplikacji odpowiedniego projektu ze źródeł, a następnie przeniesieniu plików wyjściowych do odpowiedniego folderu wskazanego w konfiguracji modułu Nginx i przeładowaniu Nginxa. Te kroki można wykonać za pomocą następujących poleceń zakładając położenie początkowe w głównym folderze plików źródłowych systemu:

```
npm --prefix Frontend/lostandfound.web ci
npm --prefix Frontend/lostandfound.web run build
cp -r Frontend/lostandfound.web/build/* ../front/
nginx -s reload
```

Moduły warstwy backendu należałyby wdrażać w analogiczny sposób, tzn. kompilować ze źródeł i uruchamiać każdy moduł osobno, ponadto należałyby uruchomić bazę danych MongoDB oraz RabbitMQ dbając przy tym o odpowiednią konfigurację sieciową każdego z modułów. Do uproszczenia sposobu uruchamiania warstwy backendu skorzystano z narzędzia Docker Compose. Pozwala ono na zdefiniowanie konfiguracji serwisów, które mają być skompilowane i uruchomione w kontenerach Dockera, a następnie uruchomienie całego rozwiązania za pomocą jednego polecenia:

```
docker-compose -f Backend/docker-compose-production.yml up --build -d
```

Do automatyzacji powyżej wymienionych zadań utworzono skrypt, który ponadto pobiera najnowszą wersję kodu z repozytorium i na jej podstawie kompiluje projekty. Pozwala on na szybkie i wygodne wdrożenie nowej wersji systemu na wcześniej skonfigurowany serwer.

---

```
# prepare code for compilation
git checkout develop
git pull

# build and copy frontend
npm --prefix Frontend/lostandfound.web ci
npm --prefix Frontend/lostandfound.web run build
rm -rf ../front/*
cp -r Frontend/lostandfound.web/build/* ../front/

# build and deploy backend to docker
docker-compose -f Backend/docker-compose-production.yml --env-file Backend/.env
up --build -d

# reload nginx
sudo nginx -s reload
```

---

#### 5.3.4. Budowanie aplikacji mobilnej

Budowanie aplikacji mobilnej jest niezależne od reszty modułów. Aby uzyskać plik .apk służący do instalacji tej aplikacji potrzebne jest posiadanie ważnego certyfikatu. Możliwe jest jego wygenerowanie z podpisem własnym za pomocą poniższej komendy:

```
keytool -genkey -v -keystore my-app-key.keystore -alias my-app-alias -keyalg RSA  
-keysize 2048 -validity 10000
```

Uzyskany certyfikat należy umieścić w folderze /Frontend/LostAndFoundMobile/android/app, a następnie skonfigurować plik /android/app/build.gradle. Następnie należy przejść do folderu /Frontend/LostAndFoundMobile i korzystając z posiadanego certyfikatu wygenerować pliki instalacyjne.

```
react-native bundle --platform android --dev false --entry-file index.js --  
bundle-output android/app/src/main/assets/index.android.bundle --assets-dest  
android/app/src/main/res/
```

Teraz można utworzyć plik .apk. W tym celu trzeba przejść do folderu /android i wykonać poniższą komendę.

```
./gradlew assembleRelease
```

Tak utworzony plik .apk znajduje się w folderze /android/app/build/outputs/apk/release/app-release.apk.

## **6. Instrukcja użytkowania**

System Lost&Found udostępnia dwie aplikacje klienckie. Aplikację webową do bezpośredniego użytku na przeglądarkach internetowych oraz aplikację do pobrania na urządzenia mobilne z systemem Android. W dalszej części tej sekcji zawarty jest opis wymagań sprzętowych oraz wszystkich funkcjonalności systemu z perspektywy użytkownika wraz z instrukcją, jak korzystać z każdej z nich.

### **6.1. Wymagania sprzętowe**

Do korzystania z aplikacji klienckich systemu Lost&Found wymagany jest stały dostęp do internetu. Ponadto dla aplikacji webowej zaleca się używanie jednej z następujących przeglądarek:

- Firefox w wersji 90 lub nowszej,
- Chrome w wersji 85 lub nowszej,
- Safari w wersji 12 lub nowszej.

W przypadku aplikacji mobilnej rekomenduje się, by spełnione były następujące wymagania sprzętowe, aby w pełni korzystać z wszystkich funkcjonalności:

- System operacyjny: Android 10.0 lub nowszy
- Pamięć: 2 GB RAM
- Przestrzeń dyskowa: 100 MB wolnego miejsca

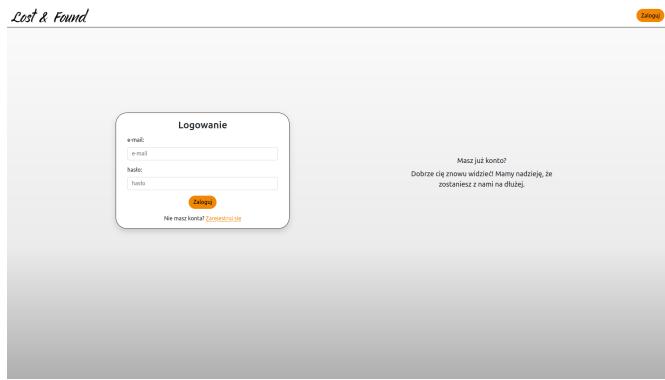
Są to minimalne wymagania na jakich aplikacje zostały przetestowane. Możliwe są również inne konfiguracje oprogramowania i sprzętu, jednakże rekommendowanym byłoby uruchamianie aplikacji na konfiguracjach nie gorszych niż te wymienione powyżej.

## 6.2. Funkcjonalności z perspektywy użytkownika

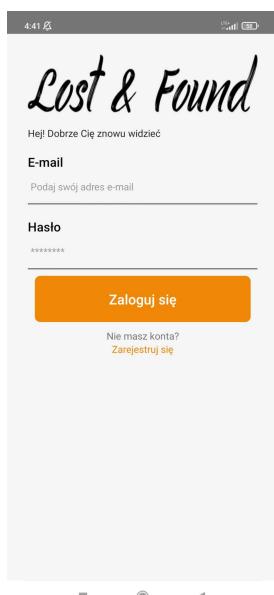
W poniższej sekcji przedstawione zostały funkcjonalności systemu Lost&Found, pokazane z perspektywy użytkownika aplikacji webowej i mobilnej. W ramach doprecyzowania, której aplikacji klienckiej dotyczy dany rysunek wykorzystano następujące skróty:

- AW - Aplikacja webowa
- AM - Aplikacja mobilna

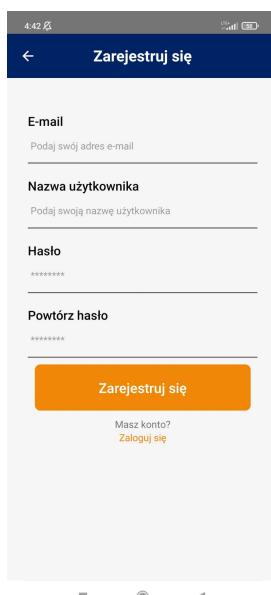
Użytkownik uruchamiając aplikację Lost&Found musi się zalogować, aby uzyskać dostęp do oferowanych funkcjonalności. W tym celu klient korzysta z poniższych widoków, które umożliwiają logowanie do aplikacji. Dodatkowo dla niezarejestrowanych użytkowników udostępniony jest proces rejestracji konta.



Rysunek 6.1: Widok logowania [AW]



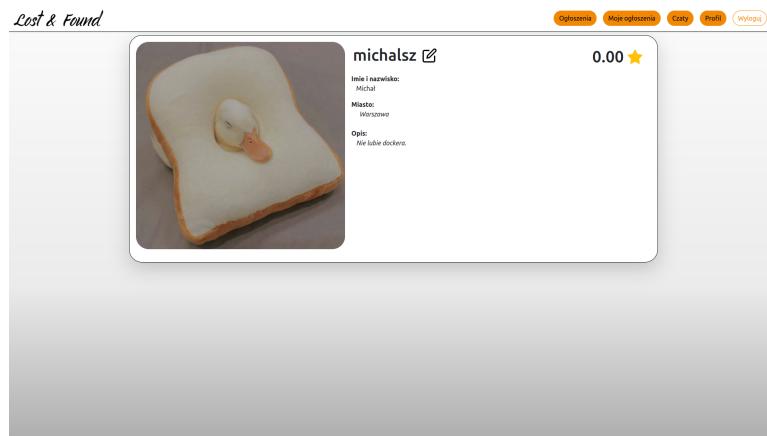
Rysunek 6.2: Widok logowania [AM]



Rysunek 6.3: Widok rejestracji [AM]

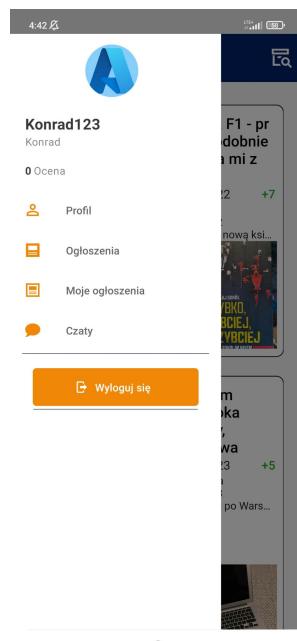
## 6.2. FUNKCJONALNOŚCI Z PERSPEKTYWY UŻYTKOWNIKA

Po zalogowaniu się do aplikacji webowej, użytkownik przekierowywany jest do strony własnego profilu. Korzystając z górnego paska nawigacji, klient może przemieszczać się pomiędzy ekranami ogłoszeń, własnych ogłoszeń, historii czatów oraz własnego profilu. Dodatkowo użytkownik ma możliwość wylogowania się z aplikacji klikając na dostępny przycisk *Wyloguj się*. Przedstawione funkcjonalności widoczne są na rysunku 6.4.



Rysunek 6.4: Widok aplikacji po poprawnym logowaniu [AW]

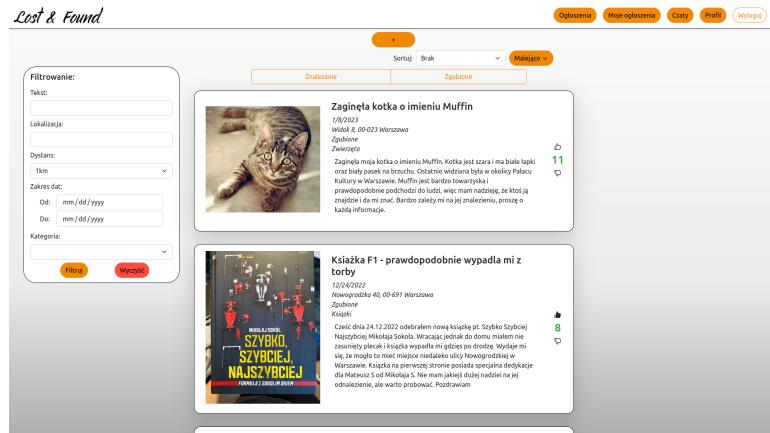
W aplikacji mobilnej użytkownik po zalogowaniu się, zostanie przekierowany do widoku ogłoszeń. Autoryzowany użytkownik ma do dyspozycji rozwijany po lewej stronie ekranu pasek menu przedstawiony na rysunku 6.5. Pozwala on na nawigację analogiczną do nawigacji dostępnej w aplikacji webowej, a także na wylogowanie się z aplikacji.



Rysunek 6.5: Widok menu [AM]

## 6. INSTRUKCJA UŻYTKOWANIA

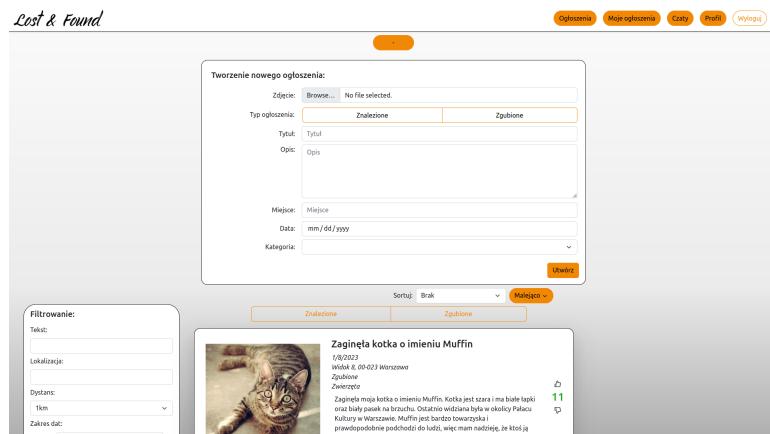
W zakładce ogłoszeń, użytkownik może przeglądać posty znalezionych i zgubionych rzeczy, oceniać je łapką w górę lub w dół. Klient ma możliwość filtrowania publikacji poprzez typ, tekstowe wyszukiwanie, datę, kategorię oraz lokalizację wraz z maksymalną odlegością ogłoszenia od podanego adresu. Dodatkowo użytkownik może posortować znalezione ogłoszenia poprzez tytuł, kategorię, datę zdarzenia, średnią ocenę, stan oraz typ ogłoszenia. Omówiony widok przedstawiony jest na rysunku 6.6.



Rysunek 6.6: Widok listy ogłoszeń [AW]

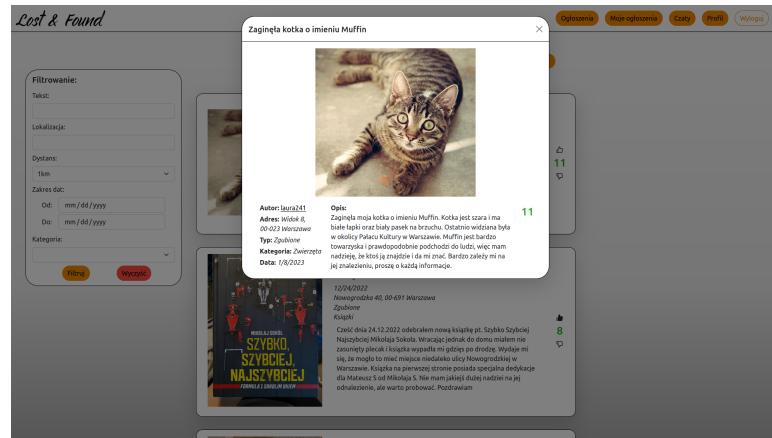
Użytkownik znajdujący się na tym ekranie, może także dodać nowe ogłoszenie wykorzystując przycisk z plusem. Proces tworzenia nowego ogłoszenia został przedstawiony na rysunku 6.7.

Klikając na jedną z publikacji, użytkownik otworzy jej szczegóły pokazane na rysunku 6.8. Stąd klient może także przejść do profilu autora ogłoszenia.



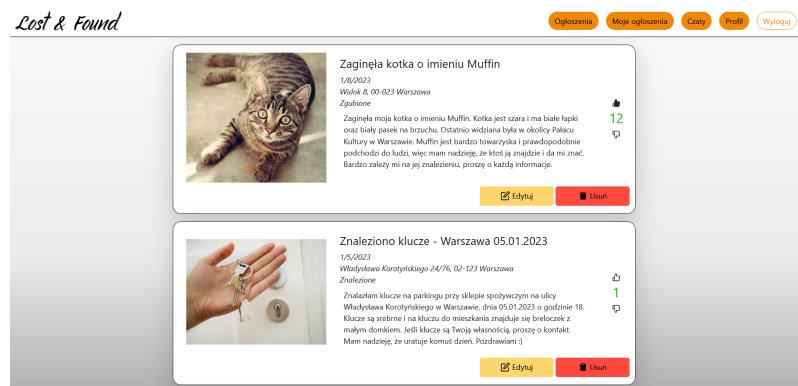
Rysunek 6.7: Widok tworzenia ogłoszeń [AW]

## 6.2. FUNKCJONALNOŚCI Z PERSPEKTYWY UŻYTKOWNIKA



Rysunek 6.8: Widok szczegółów ogłoszenia [AW]

W przypadku odwiedzenia zakładki własnych ogłoszeń, użytkownik może edytować już dodane ogłoszenia oraz je usuwać.



Rysunek 6.9: Widok własnych ogłoszeń [AW]

Na aplikacji mobilnej użytkownik przechodząc do widoku ogłoszeń może przeglądać istniejące publikacje. Klient ma możliwość wejścia w szczegóły wybranego ogłoszenia, oraz dodania własnego ogłoszenia, poprzez naciśnięcie przycisku z plusem. Klient może także przejść do wyszukiwania interesujących go ogłoszeń, klikając na ikonę w prawym górnym rogu. Widok ogłoszeń jest przedstawiony na rysunku 6.10.

Szczegóły ogłoszenia przedstawione są na rysunku 6.11. Użytkownik może polubić ogłoszenie kciukiem w górę lub w dół. Może także przejść do profilu autora publikacji lub bezpośrednio rozpoczęć z nim czat. W przypadku odwiedzenia własnego ogłoszenia, klient może je edytować lub usunąć.

## 6. INSTRUKCJA UŻYTKOWANIA



Rysunek 6.10: Widok listy ogłoszeń [AM]



Rysunek 6.11: Widok ogłoszenia [AM]

Na widoku wyszukiwania ogłoszeń pokazanym na rysunku 6.12, użytkownik może wybrać satysfakcjonujące go filtry oraz sortowanie. Wybór ten jest analogiczny do wyboru przedstawionego na aplikacji webowej.

Na rysunku 6.13 pokazano ekran tworzenia ogłoszenia. Dodatkowo analogiczny widok jest wykorzystywany przy edycji już istniejącego ogłoszenia.

The screenshot shows a search filter screen titled "Szukaj Ogłoszeń". It includes fields for "Tytuł" (Title), "Lokalizacja" (Location), "Promień" (Radius), "Data od" (From Date), "Data do" (To Date), "Kategoria" (Category), and "Typ ogłoszenia" (Ad Type).

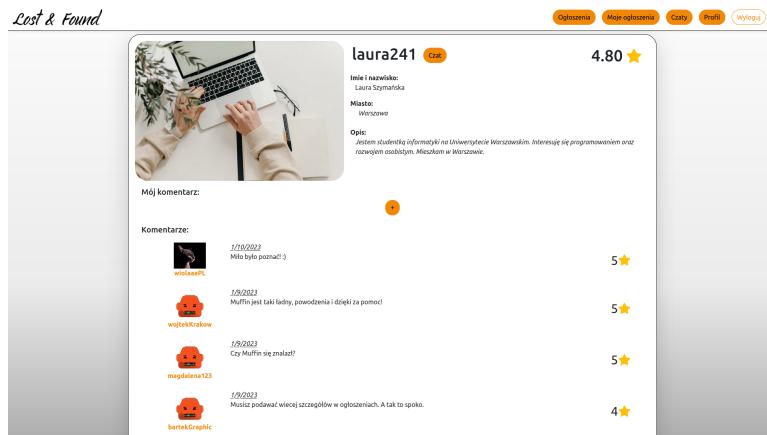
Rysunek 6.12: Widok wyszukiwania ogłoszeń [AM]

The screenshot shows a screen for creating a new ad titled "Stwórz Ogłoszenie". It includes fields for "Tytuł Ogłoszenia" (Title), "Lokalizacja" (Location), "Data" (Date), "Kategoria" (Category), "Typ ogłoszenia" (Ad Type), and "Opis" (Description). At the bottom is a large orange button labeled "Dodać ogłoszenie" (Add Ad).

Rysunek 6.13: Widok tworzenia ogłoszenia [AM]

## 6.2. FUNKCJONALNOŚCI Z PERSPEKTYWY UŻYTKOWNIKA

Użytkownicy przeglądając profil innej osoby mogą zostawić komentarz wraz z oceną w skali od 0 do 5. Mogą także rozpoczęć czat z właścicielem profilu. W przypadku odwiedzenia własnego profilu, klient może edytować swoje dane, zmienić hasło, dodać lub edytować zdjęcie profilowe.



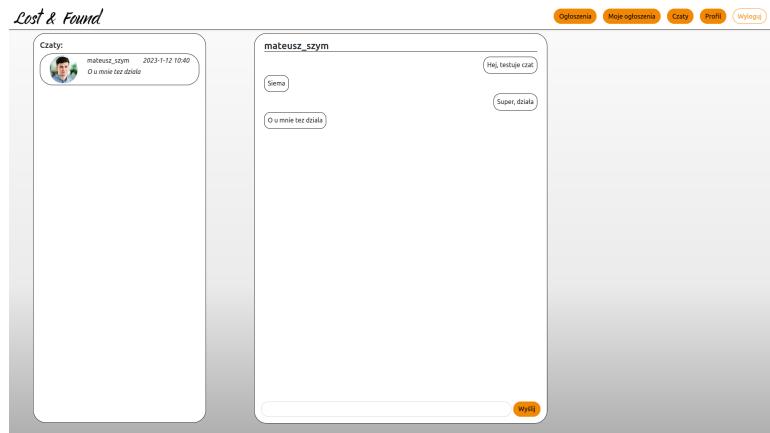
Rysunek 6.14: Widok profilu użytkownika [AW]



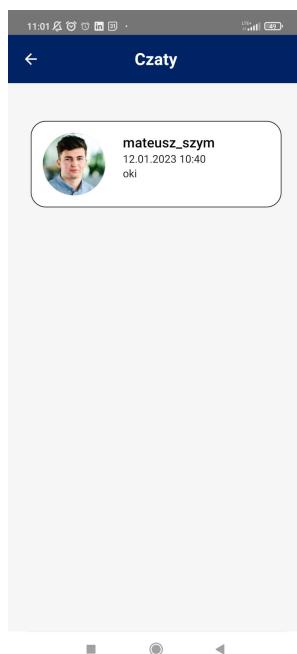
Rysunek 6.15: Widok profilu użytkownika [AM]

Użytkownik wchodząc w zakładkę czatów może przeglądać historię swoich rozmów. W przypadku nieodczytanych wiadomości, czat oznaczany jest ikoną powiadomienia. Dodatkowo liczba nieodczytyanych czatów jest widoczna na widoku menu przy zakładce *Czaty*. Klient wybierając czat z listy otworzy istniejącą rozmowę z drugim użytkownikiem, gdzie będzie miał możliwość wysłania nowej wiadomości oraz odczytania całej historii czatu.

## 6. INSTRUKCJA UŻYTKOWANIA



Rysunek 6.16: Widok czatów [AW]



Rysunek 6.17: Widok historii czatów [AM]



Rysunek 6.18: Widok czatu [AM]

## 7. Podsumowanie

Celem projektu było wypracowanie i stworzenie rozwiązania, które wprowadziłoby na rynek nowe podejście do odzyskiwania utraconych przedmiotów. Autorzy opracowali system Lost&Found będący serwisem ogłoszeniowym dla rzeczy znalezionych. Według założeń serwis powinien gromadzić ogłoszenia na temat znalezionych lub zagubionych własności i pośredniczyć w procesie ponownego łączenia ich z właścicielami.

Realizacja projektu zakończyła się sukcesem. Wszystkie zaplanowane wymagania funkcjonalne i niefunkcjonalne udało się opracować, a następnie zaimplementować i wdrożyć. Powstały system jest gotowy do użytku komercyjnego. Użytkownicy mogą korzystać z systemu zarówno za pomocą aplikacji mobilnej, jak i aplikacji webowej.

### 7.1. Wnioski

Proces realizacji zakończył się pomyślnie, jednakże zawierał drobne przeszkody. Przed przystępniem do implementacji obawialiśmy się, czy nasz brak doświadczenia w tworzeniu aplikacji mobilnych nie będzie zbyt dużym zagrożeniem. Jednakże wybór narzędzia React Native okazał się trafną decyzją. Dzięki wcześniejszej znajomości technologii React oraz języka Typescript, proces nauki i tworzenia aplikacji mobilnej nie okazał się zbyt skomplikowany. W procesie nauki bardzo przydatna była książka „React Native. Tworzenie aplikacji mobilnych w języku JavaScript” autorstwa Bonnie Eisenman, która jest przeznaczona dla osób przystępujących do nauki React Native i znających już podstawy biblioteki React [1].

Inną możliwością jaką teraz zauważamy, byłoby porzucenie aplikacji mobilnej na rzecz rozwoju w pełni responsywnej aplikacji webowej, wykorzystując w pełni potencjał biblioteki Bootstrap - byłaby ona wygodna w użyciu nie tylko na komputerach, ale także na telefonach. Zmniejszyłoby to potrzebny nakład pracy oraz ułatwiło ujednolicanie interfejsu aplikacji. Zaoszczędzone w ten sposób zasoby moglibyśmy przeznaczyć na rozwój dodatkowych funkcjonalności.

Dodatkowo rozwój warstwy backendu jako struktury mikroserwisów nie wnosi obecnie znaczących wartości do funkcjonowania systemu. Zauważamy, że takie rozwiązanie znacząco rozszerzyło

wymaganą pracę do utworzenia w pełni działającego systemu, jednak obecnie nie wykorzystywany jest jego pełny potencjał. Architektura mikroserwisów uwiadocznia swoje zalety dopiero podczas dalszego rozwoju i rozrostu systemu.

## 7.2. Perspektywy

Mimo iż powstały system Lost&Found jest w pełni funkcjonalny i gotowy do wyjścia na rynek, przy ewentualnym dalszym rozwoju, wciąż zauważamy pewne potrzeby.

Obecny model wdrożenia stanowi pewne ograniczenie - szacunkowa liczba użytkowników mogących korzystać jednocześnie z serwisu w sposób satysfakcjonujący nie przekracza 500. Wynika to z uruchomienia całego systemu z wykorzystaniem tylko jednego serwera. Wraz ze wzrostem liczby użytkowników należałoby rozważyć i zaimplementować rozwiązania dotyczące skalowania. Pierwszym krokiem powinno być przeniesienie poszczególnych mikroserwisów na osobne maszyny. Wykorzystałoby to w pełni zalety zastosowanej architektury. Następnym krokiem mogłoby być skalowanie zarówno poziomo - równoległe uruchomienie wielu instancji jednego serwisu wraz z systemem równoważącym obciążenie, jak i pionowe - wykorzystanie maszyn o lepszych parametrach. Przedstawione rozwiązania zwiększyłyby koszty utrzymania serwisu, które należałyby rekompensować. Jednym z rozwiązań mogłoby być wprowadzanie reklam w aplikacjach klienckich. Inny sposób generowania zysków, można osiągnąć wprowadzając możliwości płatnego promowania ogłoszeń.

Dodatkowo dla rozszerzenia bazy ogłoszeń możliwym byłoby nawiązanie współpracy z lokalnymi biurami rzeczy znalezionych. Pracownicy tych biur mogliby korzystać z serwisu Lost&Found, publikując tam dostarczone im przedmioty. Taka współpraca znacząco zwiększyłaby liczbę ogłoszeń systemu, co pociągało by za sobą rozwój serwisu.

W dalszej perspektywie słusznym mogłoby być również rozszerzenie aplikacji klienckich o widoki zawierające mapy. Pozwoliłoby to na bardziej przyjazne użytkownikowi przeglądanie list ogłoszeń. Zauważamy również potrzebę implementacji prostego modułu, który mógłby powiadać użytkowników o nowych wiadomościach i ogłoszeniach, pomagając w poszukiwaniu utraconej własności. Ponadto możliwość komentowania ogłoszeń, przykładowo tych dotyczących zagubionych zwierząt, zdecydowanie mogłoby wnieść wartość dodatnią do systemu.

## Bibliografia

- [1] Bonnie Eisenman, *React Native. Tworzenie aplikacji mobilnych w języku JavaScript.*, Helion, 2018.
- [2] Natan Gil, *Przewodnik po mediach społecznościowych 2022 – analiza trendów w Polsce*, emedia, 2022, <https://emedia.pl/blog/przewodnik-po-mediach-społecznościowych-2022-analiza-trendow-w-polsce>.
- [3] Robert Marshall, *Arrange, Act and Assert Pattern: The Three A's of Unit Testing*, 2022, <https://robertmarshall.dev/blog/arrange-act-and-assert-pattern-the-three-as-of-unit-testing/>.
- [4] Rafał Motyl, *Kto pyta, ten może znaleźć*, Urząd miasta stołecznego Warszawy, 2022, <https://um.warszawa.pl/-/kto-pyta-ten-moze-znalezc>.
- [5] Carl Rippon, *ASP.NET Core 5 and React*, Packt Publishing Ltd, Birmingham, 2021.
- [6] Oleksandr Yeremenko, *Documenting Software Requirements: How to Do It Right?*, Andersen, 2022, <https://andersenlab.com/blueprint/ba-artifacts-in-software-development-ins-outs>.
- [7] Wojciech Zawistowski, *3 typy testów, które musisz zrozumieć, by robić efektywne TDD*, eSky, 2014, <https://it.esky.pl/programowanie/3-typy-testow-które-musisz-zrozumieć-by-robić-efektywne-tdd/>.

## **Spis rysunków**

3.1	Poglądowy schemat architektury systemu Lost&Found . . . . .	20
3.2	Konfiguracja zasobów magazynu obiektów systemu Lost&Found . . . . .	25
5.1	Schemat konfiguracji wdrożenia . . . . .	40
6.1	Widok logowania [AW] . . . . .	46
6.2	Widok logowania [AM] . . . . .	46
6.3	Widok rejestracji [AM] . . . . .	46
6.4	Widok aplikacji po poprawnym logowaniu [AW] . . . . .	47
6.5	Widok menu [AM] . . . . .	47
6.6	Widok listy ogłoszeń [AW] . . . . .	48
6.7	Widok tworzenia ogłoszeń [AW] . . . . .	48
6.8	Widok szczegółów ogłoszenia [AW] . . . . .	49
6.9	Widok własnych ogłoszeń [AW] . . . . .	49
6.10	Widok listy ogłoszeń [AM] . . . . .	50
6.11	Widok ogłoszenia [AM] . . . . .	50
6.12	Widok wyszukiwania ogłoszeń [AM] . . . . .	50
6.13	Widok tworzenia ogłoszenia [AM] . . . . .	50
6.14	Widok profilu użytkownika [AW] . . . . .	51
6.15	Widok profilu użytkownika [AM] . . . . .	51
6.16	Widok czatów [AW] . . . . .	52
6.17	Widok historii czatów [AM] . . . . .	52
6.18	Widok czatu [AM] . . . . .	52

## **Spis tabel**

1.1	Analiza Ryzyka . . . . .	14
2.1	Wymagania dotyczące przeglądania ogłoszeń. . . . .	15
2.2	Wymagania dotyczące rejestracji, logowania i konta użytkownika. . . . .	16
2.3	Wymagania dotyczące tworzenia i edycji ogłoszeń. . . . .	16
2.4	Wymagania dotyczące szczegółów ogłoszeń. . . . .	17
2.5	Wymagania dotyczące profili innych użytkowników. . . . .	17
2.6	Wymagania dotyczące czatu. . . . .	17
2.7	Lista wymagań niefunkcjonalnych systemu. . . . .	18