



# POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

Wydział Informatyki

Filia w Gdańsku

**Langmesser Adam**

Nr albumu s27119

Nazwa specjalizacji: Aplikacje Internetowe

**Redosz Mateusz**

Nr albumu s27094

Nazwa specjalizacji: Aplikacje Internetowe

**Oziemczuk Stanisław**

Nr albumu s26982

Nazwa specjalizacji: Aplikacje Internetowe

**Badek Kacper**

Nr albumu s29168

Nazwa specjalizacji: Aplikacje Internetowe

## Aplikacja webowa: spoty-na-drony.pl

Rodzaj pracy

inżynierska

Imię i nazwisko promotora

mgr Adam Urbanowicz

Gdańsk, miesiąc, 2100 obrony

**Streszczenie:** Celem niniejszej pracy było stworzenie w pełni funkcjonalnej i działającej aplikacji internetowej pozwalającej na szybkie wyszukiwanie spotów w okolicy oraz dzielenie się zdjęciami, filmami oraz doświadczeniem z innymi użytkownikami. W ramach pracy stworzono system składający się z trzech komponentów: [Frontendu](#), [Backendu](#) oraz bazy-danych. Aplikacja internetowa została wykonana przy pomocy [Frameworka](#) React w językach Javascript oraz Typescript, do stylu został użyty Tailwind. Serwis backendowy został stworzony w języku Java oraz biblioteki Spring Boot. Baza danych to PostgreSQL.

Komunikacja między komponentami odbywała się zgodnie ze standardem REST. Projekt został zrealizowany w podejściu ewolucyjno-przyrostowym z elementami Kanban.

**Słowa kluczowe:** — brak —



# POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

## Karta projektu

<b>Temat projektu:</b> Aplikacja webowa: spoty-na-drony.pl <b>Temat projektu po angielsku:</b> Web application: spoty-na-drony.pl	<b>Akronim:</b> Merkury <b>Data ustalenia tematu</b> 2023-10-10
<b>Promotor:</b>  mgr Adam Urbanowicz	<b>Konsultanci:</b>  1. — brak —
<b>Cele projektu:</b> Stworzenie w pełni funkcjonalnej aplikacji internetowej do rozwijania hobby(latania dronem).	
<b>Rezultaty projektu:</b> Aplikacja Internetowa, Dokumentacja Interaktywna mapa z wyświetlanymi spotami oraz pogodą. Zaawansowana wyszukiwarka spotów. Forum do dzielenia się informacjami na temat dronów. Chat jednoosobowy oraz grupowy. Konto użytkownika z możliwością zapisania ulubionych spotów.	
<b>Miary sukcesu:</b> Gotowa do wdrożenia aplikacja. Realizacja w terminie zgodnym z wymaganiami.	
<b>Ograniczenia:</b> Budżetowe: brak środków na wdrożenie. Zawodowe: brak doświadczenia. Czasowe: trzy semestry (09.2024 - 02.2026). Ludzkie: czteroosobowy zespół.	

Wykonawcy	Numer al- bumu	Specjalizacja	Tryb studiów
Langmesser Adam	s27119	Aplikacje Internetowe	Stacjonarny
Redosz Mateusz	s27094	Aplikacje Internetowe	Stacjonarny
Oziemczuk Stanisław	s26982	Aplikacje Internetowe	Stacjonarny
Badek Kacper	s29168	Aplikacje Internetowe	Stacjonarny

<b>Data ukończenia projektu:</b> 1 grudnia 2025	<b>Recenzent:</b> dr Elżbieta Puźniakowska-Gałuch
--	--

# Spis treści

<b>1</b>	<b>Wstęp</b>	<b>6</b>
1.1	O projekcie . . . . .	6
1.2	Cel i zakres prac . . . . .	6
1.3	Geneza pomysłu . . . . .	6
<b>2</b>	<b>Opis problemu</b>	<b>7</b>
2.1	Rich picture . . . . .	7
2.2	Udziałowcy . . . . .	7
2.3	Istniejące rozwiązania . . . . .	9
2.4	Wizja rozwiązania . . . . .	9
2.5	Aspekty społeczne i biznesowe . . . . .	9
2.5.1	Aspekty społeczne . . . . .	9
2.5.2	Aspekty biznesowe . . . . .	9
<b>3</b>	<b>Planowanie</b>	<b>10</b>
3.1	Metodologia pracy . . . . .	10
3.1.1	Przegląd rozważanych podejść . . . . .	10
3.1.2	Odrzucone podejścia . . . . .	10
3.1.3	Wybrane podejście: Disciplined Agile Delivery (Lean Life Cycle) . . . . .	11
3.1.4	Narzędzia i komunikacja . . . . .	11
3.1.5	Podział ról w zespole . . . . .	12
3.2	Harmonogram projektu . . . . .	12
3.3	Technologie i narzędzia . . . . .	14
3.3.1	Technologie . . . . .	14

3.3.2	Narzędzia . . . . .	14
3.4	Zasoby i ograniczenia . . . . .	17
3.4.1	Zasoby . . . . .	17
3.4.2	Ograniczenia . . . . .	17
3.4.3	Usługi zewnętrzne . . . . .	18
3.5	Analiza ryzyka . . . . .	20
<b>4</b>	<b>Analiza wymagań</b>	<b>21</b>
4.1	Przypadki użycia . . . . .	21
4.1.1	Aktorzy . . . . .	21
4.1.2	Diagramy przypadków użycia . . . . .	23
4.1.3	Scenariusz przypadków użycia . . . . .	32
4.2	Wymagania ogólne i dziedzinowe . . . . .	32
4.3	Wymagania funkcjonalne . . . . .	32
4.3.1	Funkcjonalności dla mapy . . . . .	32
4.3.2	Funkcjonalności dla chatu . . . . .	32
4.3.3	Funkcjonalności dla forum . . . . .	32
4.3.4	Funkcjonalności dla konta użytkownika . . . . .	32
4.3.5	Funkcjonalności dla logowania i rejestracji . . . . .	42
4.3.6	Funkcjonalności dla wyszukiwarki spotów . . . . .	43
4.3.7	Funkcjonalności dla motywu . . . . .	45
4.4	Wymagania pozafunkcjonalne . . . . .	47
4.5	Wymagania interfejs z otoczeniem . . . . .	47
4.6	Wymagania na środowisko docelowe . . . . .	47
<b>5</b>	<b>Projekt</b>	<b>48</b>
5.1	Wzorce projektowe . . . . .	48
5.2	Architektura systemu . . . . .	48
5.2.1	Diagram architektury . . . . .	49
5.2.2	Komponenty systemu . . . . .	51
5.3	Projekt bazy danych . . . . .	52
5.3.1	Model danych . . . . .	52

5.3.2	Diagram ERD . . . . .	52
5.4	Architektura interfejsu użytkownika . . . . .	52
5.4.1	Projekt strony głównej . . . . .	52
5.4.2	Projekt panelu logowania . . . . .	52
5.4.3	Projekt mapy . . . . .	52
5.4.4	Projekt chatu . . . . .	52
5.4.5	Projekt forum . . . . .	52
5.4.6	Projekt konta użytkownika . . . . .	52
<b>6</b>	<b>Przebieg realizacji projektu</b>	<b>53</b>
6.1	Sprint 1 . . . . .	53
6.2	Sprint 2 . . . . .	53
<b>7</b>	<b>Realizacja Projektu</b>	<b>54</b>
7.1	Implementacja backendu . . . . .	54
7.1.1	Struktura projektu . . . . .	54
7.1.2	Endpointy systemu . . . . .	57
7.1.3	Integracja z bazą danych . . . . .	100
7.1.4	Obsługa uwierzytelnienia . . . . .	102
7.1.5	Konteneryzacja . . . . .	102
7.2	Implementacja frontendu . . . . .	102
7.2.1	Struktura aplikacji . . . . .	102
7.2.2	Zarządzanie stanem i przepływ danych . . . . .	107
7.2.3	Integracja i komunikacja z backendem . . . . .	110
7.2.4	Style . . . . .	113
7.2.5	Wyszukiwarka spotów . . . . .	117
7.2.6	Mapa . . . . .	124
7.2.7	Chat . . . . .	124
7.2.8	Forum . . . . .	124
7.2.9	Konto użytkownika . . . . .	124
7.2.10	Panel logowania . . . . .	124
7.3	Implementacja CI/CD . . . . .	124

<b>8 Testy</b>	<b>125</b>
8.1 Testy jednostkowe . . . . .	125
8.2 Testy integracyjne . . . . .	125
8.3 Testy E2E . . . . .	125
8.4 Wyniki testów i wnioski . . . . .	125
<b>9 Prezentacja systemu</b>	<b>126</b>
9.1 Strona główna . . . . .	126
9.2 Strona mapy . . . . .	126
9.3 Strona chatu . . . . .	126
9.4 Strona forum . . . . .	126
9.5 Panel logowania . . . . .	126
9.6 Panel konta użytkownika . . . . .	126
<b>10 Nakład pracy</b>	<b>127</b>
10.1 Ogólny nakład pracy . . . . .	127
10.2 Indywidualne nakłady pracy . . . . .	127
10.2.1 Adam Langmesser . . . . .	127
10.2.2 Mateusz Redosz . . . . .	127
10.2.3 Stanisław Oziemczuk . . . . .	130
10.2.4 Kacper Badek . . . . .	130
<b>11 Podsumowanie</b>	<b>131</b>
11.1 Osiągnięte rezultaty . . . . .	131
11.2 Napotkane wyzwania . . . . .	131
11.3 Plany na przyszłość . . . . .	131
<b>12 Słownik pojęć i skrótów</b>	<b>132</b>
<b>Spis tabel</b>	<b>140</b>
<b>Bibliografia</b>	<b>143</b>
<b>Załączniki</b>	<b>145</b>

# Rozdział 1

## Wstęp

1.1 O projekcie

1.2 Cel i zakres prac

1.3 Geneza pomysłu



## Rozdział 2

### Opis problemu

#### 2.1 Rich picture

#### 2.2 Udziałowcy

KARTA UDZIAŁOWCA	
<b>Identyfikator:</b>	U01
<b>Nazwa udziałowca:</b>	Zespół projektowy
<b>Opis:</b>	Zespół czterech studentów odpowiedzialnych za analizę, projekt, implementację, testy oraz dokumentację systemu.
<b>Typ:</b>	ożywiony, bezpośredni
<b>Perspektywa:</b>	Techniczna, wykonawcza.
<b>Ograniczenia:</b>	Ograniczone zasoby czasowe i doświadczenie komercyjne.
<b>Powiązane wymagania:</b>	Wymagania funkcjonalne i techniczne systemu, możliwość realizacji w ramach projektu dyplomowego.

**Tabela 2.1:** Karta udziałowca: Zespół projektowy

KARTA UDZIAŁOWCA	
<b>Identyfikator:</b>	UO2
<b>Nazwa udziałowca:</b>	Promotor
<b>Opis:</b>	Osoba nadzorująca przebieg projektu, weryfikująca poprawność merytoryczną i zgodność z wymaganiami uczelni.
<b>Typ:</b>	ożywiony, pośredni
<b>Perspektywa:</b>	Merytoryczna, formalna, jakościowa.
<b>Ograniczenia:</b>	Nie odpowiada za implementację; rekomenduje, opiniuje i zatwierdza.
<b>Powiązane wymagania:</b>	Czytelna dokumentacja, zgodność z wytycznymi kierunku oraz odpowiedni poziom techniczny rozwiązania.

**Tabela 2.2:** Karta udziałowca: Promotor

KARTA UDZIAŁOWCA	
<b>Identyfikator:</b>	UO3
<b>Nazwa udziałowca:</b>	<a href="#">Droniarze</a>
<b>Opis:</b>	Główna grupa docelowa systemu – osoby latające dronami rekreacyjnie lub półprofesjonalnie, szukające miejsc do lotów i wymiany doświadczeń.
<b>Typ:</b>	ożywiony, bezpośredni
<b>Perspektywa:</b>	Użytkownik końcowy: prostota obsługi, rzetelne informacje o spotach, wygodne dzielenie się treściami.
<b>Ograniczenia:</b>	Brak wpływu na architekturę techniczną systemu; oczekują intuicyjnego interfejsu.

<b>Powiązane wymagania:</b>	Lista spotów, informacje o ograniczeniach prawnych, oceny i komentarze, dodawanie treści oraz podstawowe funkcje społecznościowe.
-----------------------------	---

**Tabela 2.3:** Karta udziałowca: [Droniarze](#)

## 2.3 Istniejące rozwiązania

## 2.4 Wizja rozwiązania

## 2.5 Aspekty społeczne i biznesowe

### 2.5.1 Aspekty społeczne

### 2.5.2 Aspekty biznesowe

# Rozdział 3

## Planowanie

### 3.1 Metodologia pracy

#### 3.1.1 Przegląd rozważanych podejść

Przy wyborze metodologii pracy rozważono trzy podejścia do prowadzenia projektu informatycznego:

- klasyczny Agile (w praktyce: Scrum),
- model kaskadowy (Waterfall),
- [Disciplined Agile Delivery - Lean Life Cycle](#).

#### 3.1.2 Odrzucone podejścia

**„Klasyczny Agile” (Scrum).** Mimo elastyczności i popularności zakłada pracę w iteracjach 2–4 tygodni oraz stały zestaw ceremonii (planowanie, przegląd, retrospektywa). Ze względu na nierównomierną dostępność zasobów w kolejnych miesiącach studiów nie zapewniono możliwości utrzymania stałej kadencji sprintów, dlatego z podejścia zrezygnowano.

**Model kaskadowy (Waterfall).** Przewiduje sekwencyjne przechodzenie przez z góry określone etapy i ogranicza bieżącą weryfikację wymagań w trakcie prac deweloperskich. W projekcie wymagano możliwości częstych rewizji założeń oraz

wprowadzania istotnych zmian w docelowej wizji rozwiązania; dlatego z podejścia zrezygnowano.

### 3.1.3 Wybrane podejście: Disciplined Agile Delivery (Lean Life Cycle)

Podjęto decyzję o zastosowaniu **Disciplined Agile Delivery** [1] w wariancie **Lean Life Cycle** [2], ponieważ podejście to łączy pożądane cechy Agile i Waterfall, a jednocześnie eliminuje stałe sprinty na rzecz pracy w ciągłym przepływie.

#### Kluczowe argumenty wyboru:

- **Brak sprintów.** Zastosowano przepływ ciągły, co pozwala dopasować tempo do zmiennej dostępności zespołu i unikać sztucznego „domykania” iteracji.
- **Rozbudowana faza startowa.** Na początku przewidziano większy wysiłek planistyczny: doprecyzowanie zakresu, wstępna wizja architektury, identyfikacja ryzyk, plan publikacji oraz kryteria jakości – bez zamrażania szczegółów.
- **Ciągła weryfikacja wymagań.** W trakcie realizacji przewidziano bieżące doprecyzowywanie backlogu, regularny feedback promotora oraz możliwość korygowania kierunku bez kosztów „przeskakiwania” między fazami.
- **Praktyki Lean i koncentracja na wartości.** Priorytetyzacja wartości biznesowej, wizualizacja pracy, małe partie dostaw.
- **Lekka governance i kamienie milowe.** Zastosowano lekkie mechanizmy nadzoru (peer review, prezentacje postępów) zapewniające przejrzystość bez nadmiernej biurokracji.

### 3.1.4 Narzędzia i komunikacja

Do zarządzania zadaniami zastosowana została **Jira** (monitorowanie postępu prac oraz ewidencja zadań członków zespołu). Komunikację w zespole zaplanowano w

formie regularnych spotkań oraz asynchronicznie z wykorzystaniem **Discorda** oraz **Messengera**.

### 3.1.5 Podział ról w zespole

- Adam - fullstack developer, lider zespołu
- Stanisław - fullstack developer
- Kacper - fullstack developer
- Mateusz - fullstack developer

Każdy z członków zespołu uczestniczy również w przygotowaniu dokumentacji.

## 3.2 Harmonogram projektu

W poniższym harmonogramie przedstawiono plan prac nad poszczególnymi częściami projektu, rozłożony na miesiące.

### Rok 2024

**Czerwiec**     • Zebranie zespołu.

- Rozważenie potencjalnych pomysłów.

**Lipiec**     • Wybór technologii.

- Wstępne założenia architektoniczne.

**Sierpień**     • *(do uzupełnienia)*

- *(do uzupełnienia)*

**Wrzesień**     • *(do uzupełnienia)*

- *(do uzupełnienia)*

**Październik**     • *(do uzupełnienia)*

- *(do uzupełnienia)*

**Listopad**     • *(do uzupełnienia)*

- *(do uzupełnienia)*

**Grudzień**     • *(do uzupełnienia)*

- *(do uzupełnienia)*

## **Rok 2025**

**Styczeń**     • *(do uzupełnienia)*

- *(do uzupełnienia)*

**Luty**     • *(do uzupełnienia)*

- *(do uzupełnienia)*

**Marzec**     • *(do uzupełnienia)*

- *(do uzupełnienia)*

**Kwiecień**     • *(do uzupełnienia)*

- *(do uzupełnienia)*

**Maj**     • *(do uzupełnienia)*

- *(do uzupełnienia)*

**Czerwiec**     • *(do uzupełnienia)*

- *(do uzupełnienia)*

**Lipiec**     • *(do uzupełnienia)*

- *(do uzupełnienia)*

**Sierpień**     • *(do uzupełnienia)*

- *(do uzupełnienia)*

**Wrzesień**     • *(do uzupełnienia)*

- *(do uzupełnienia)*

**Październik**    • *(do uzupełnienia)*

- *(do uzupełnienia)*

**Listopad**    • *(do uzupełnienia)*

- *(do uzupełnienia)*

**Grudzień**    • *(do uzupełnienia)*

- *(do uzupełnienia)*

**Rok 2026**

**Styczeń**    • *(do uzupełnienia)*

- *(do uzupełnienia)*

### 3.3 Technologie i narzędzia

Do realizacji projektu wykorzystano wiele technologii oraz narzędzi informatycznych. Przy wyborze technologii kierowaliśmy się ich popularnością, dostępnością dokumentacji oraz artykułów, a także łatwością użycia. Narzędzia zostały dopasowane do wybranych technologii i specyfikacji zadań. Poniżej przedstawiono opis wybranych opcji.

#### 3.3.1 Technologie

#### 3.3.2 Narzędzia

Do niektórych płatnych narzędzi otrzymano bezpłatny dostęp za pośrednictwem uczelni, w innych istniała możliwość założenia konta edukacyjnego, które oferowało dostęp do wszystkich funkcji narzędzia. Gdy żadna z wymienionych opcji nie była udostępniona, wybierano rozwiązania darmowe.

- **IntelliJ IDEA Ultimate**

Jest to [IDE](#) od firmy JetBrains. Dzięki licznie dostępnym pluginom oferuje obsługę wielu języków programowania oraz innych składni. Pozwala również



na integrację z repozytorium. Używano go do programowania zarówno [frontendu](#), jak i [backendu](#) oraz tworzenia dokumentacji w LaTeX.

- **Docker Desktop**

To narzędzie do zarządzania obrazami, kontenerami oraz wolumenami Docker. Zawiera w sobie również silnik tej technologii. Wykorzystywano je do lokalnego uruchamiania bazy danych oraz serwisu do cachowania.

- **One Drive**

Usługa dysku chmurowego oferowana przez firmę Microsoft. Przechowywano tam dokumenty oraz obrazy diagramów.

- **Azure Blob Storage**

To rozwiązanie chmurowe Microsoft, służące do bezpiecznego przechowywania dużej ilości danych nieustrukturyzowanych, takich jak pliki multimedialne, dokumenty czy kopie zapasowe. Dane są dostępne poprzez interfejs [REST API](#) usługi Azure Storage. Wykorzystywano je do przechowywania zdjęć profilowych użytkownika oraz multimedii (zdjęcia i filmy) ze [spotów](#) i forum.

- **Jira**

To narzędzie firmy Atlassian do zarządzania pracami nad projektem w metodach zwinnych. Do [Backlogu](#) wpisywano zadania, a na [tablicy Kanbanowej](#) rejestrowano ich statusy oraz poświęcony czas.

- **GitHub**

Zdalne repozytorium służące do przechowywania i wersjonowania kodu aplikacji. Zamieszczono tam kod naszego projektu. Do każdego zadania tworzone osobną gałąź z właściwą nazwą, a po zakończeniu prac przeprowadzano [review kodu](#). Następnie łączono ją do głównej gałęzi deweloperskiej.

- **GitHub Actions**

To narzędzie do implementacji procesów [CI/CD](#) na platformie GitHub, które umożliwiają automatyczne testowanie lub wdrażanie kodu. Uruchamiają się w reakcji na różne operacje w repozytorium, na przykład przesłanie zmian na wybraną gałąź. Stosowano je do automatycznego testowania i budowania projektu po każdorazowym wprowadzeniu zmian.

- **GitHub Copilot**

To narzędzie sztucznej inteligencji będące asystentem programisty. W projekcie analizuje plik oraz pliki powiązane. Wykorzystywano go podczas [review kodu](#). Copilot skanuje wszystkie pliki i w komentarzach opisuje sugerowane zmiany lub potencjalne błędy.

- **Discord**

Darmowa platforma komunikacyjna. Umożliwia udostępnienie obrazu z ekranu, komunikację głosową oraz tekstową, jak i również przesyłanie plików. Stosowano go do spotkań, na których omawiano sprawy dotyczące projektu.

- **Messenger**

Komunikator będący usługą Facebooka. Daje możliwość tworzenia czatów grupowych lub prywatnych, a także udostępniania plików. Używano go do ustalania spotkań na Discordzie oraz szybkiej komunikacji.

- **Postman**

To narzędzie służące do testowania endpointów [API](#). Pozwala grupować zapytania w kolekcje, wysyłać ich różne typy oraz analizować odpowiedzi z serwera. Wykorzystywano go do testowania stworzonych endpointów oraz debugowania.

- **Figma**

Narzędzie chmurowe do projektowania interfejsów użytkownika ([UI](#)). Umożliwia zespołowe tworzenie w pełni interaktywnych prototypów. Wykonano w nim projekty ekranów naszej aplikacji.

- **Visual Paradigm**

To narzędzie do tworzenia różnych diagramów stosowanych w inżynierii oprogramowania, takich jak [UML](#)( [3]) czy [BPMN](#)( [4]). Zrobiono w nim diagram przypadków użycia.

- **Xmind**

Narzędzie służące do tworzenia mapy myśli. Wykorzystano je w celu lepszego zrozumienia problemów poprzez przeniesienie ich na diagram.

## 3.4 Zasoby i ograniczenia

### 3.4.1 Zasoby

- **Specjalizacja członków zespołu** — wszyscy członkowie zespołu projektowego specjalizują się w aplikacjach internetowych.
- **Dostęp do przedstawiciela grupy docelowej** — jeden z członków zespołu (Adam) jest [droniarzem foto/video](#).
- **Status studenta** — fakt bycia studentem zapewnia dostęp do wersji premium wielu usług (Figma Education, GitHub PRO).
- **Oprogramowanie zapewniane przez PJATK** - uczelnia zapewnia dostęp do pakietu JetBrains oraz usług firmy Microsoft (OneDrive).

### 3.4.2 Ograniczenia

- **Ograniczenia czasowe** — projekt jest ograniczony harmonogramem akademickim i terminem oddania pracy dyplomowej, co wymagało wysokiego tempa realizacji oraz sprawnej komunikacji w zespole.
- **Ograniczenia budżetowe** — projekt nie posiada finansowania i w związku z tym korzystano z rozwiązań darmowych oraz open source.

### 3.4.3 Usługi zewnętrzne

Niniejszy rozdział zawiera spis zewnętrznych [API](#) oraz usług użytych w projekcie.

KARTA USŁUGI ZEWNĘTRZNEJ	
<b>Identyfikator:</b>	UZ1
<b>Nazwa:</b>	GitHub Actions (CI) [5]
<b>Opis:</b>	Uruchomienia pipeline'ów CI/CD dla repozytorium GitHub.
<b>Limit:</b>	3000 min/mies.

**Tabela 3.1:** Usługa zewnętrzna: GitHub Actions (CI)

KARTA USŁUGI ZEWNĘTRZNEJ	
<b>Identyfikator:</b>	UZ2
<b>Nazwa:</b>	Azure Blob Storage [6]
<b>Opis:</b>	Magazyn plików (m.in. zdjęcia spotów, załączniki z czatu).
<b>Limit:</b>	1 GB/mies.

**Tabela 3.2:** Usługa zewnętrzna: Azure Blob Storage

KARTA USŁUGI ZEWNĘTRZNEJ	
<b>Identyfikator:</b>	UZ3
<b>Nazwa:</b>	Mailtrap [7]
<b>Opis:</b>	Środowisko testowe SMTP oraz Email API do wysyłki maili.
<b>Limit:</b>	150 maili/dzień

**Tabela 3.3:** Usługa zewnętrzna: Mailtrap

KARTA USŁUGI ZEWNĘTRZNEJ	
<b>Identyfikator:</b>	UZ4
<b>Nazwa:</b>	LocationIQ [8]
<b>Opis:</b>	Geokodowanie adresu przy dodawaniu nowych spotów.
<b>Limit:</b>	5 000 zapytań/dzień

**Tabela 3.4:** Usługa zewnętrzna: LocationIQ

KARTA USŁUGI ZEWNĘTRZNEJ	
<b>Identyfikator:</b>	UZ5
<b>Nazwa:</b>	Google Maps (Maps URLs) [9]
<b>Opis:</b>	Otwieranie nawigacji w aplikacji Map Google (deep link/URL).
<b>Limit:</b>	Brak limitu w ramach dokumentowanego sposobu użycia.

**Tabela 3.5:** Usługa zewnętrzna: Google Maps (Maps URLs)

KARTA USŁUGI ZEWNĘTRZNEJ	
<b>Identyfikator:</b>	UZ6
<b>Nazwa:</b>	OpenFreeMap [10]
<b>Opis:</b>	Publiczny serwer kafelków do renderu mapy na froncie.
<b>Limit:</b>	30 000 zapytań/mies.

**Tabela 3.6:** Usługa zewnętrzna: OpenFreeMap

KARTA USŁUGI ZEWNĘTRZNEJ	
<b>Identyfikator:</b>	UZ7
<b>Nazwa:</b>	Open-Meteo [11]
<b>Opis:</b>	Prognozy pogody wyświetlane dla spotów.
<b>Limit:</b>	10 000 zapytań/dzień

**Tabela 3.7:** Usługa zewnętrzna: Open-Meteo

KARTA USŁUGI ZEWNĘTRZNEJ	
<b>Identyfikator:</b>	UZ8
<b>Nazwa:</b>	Tenor GIF API [12]
<b>Opis:</b>	Wyszukiwanie GIF-ów w czacie.
<b>Limit:</b>	1 zapytanie na sekundę; brak ogólnego limitu dziennego.

**Tabela 3.8:** Usługa zewnętrzna: Tenor GIF API

KARTA USŁUGI ZEWNĘTRZNEJ	
<b>Identyfikator:</b>	UZ9
<b>Nazwa:</b>	Where the ISS at? [13]
<b>Opis:</b>	HTTP API z bieżącą pozycją satelity, używane pomocniczo.
<b>Limit:</b>	1 zapytanie na sekundę; brak ogólnego limitu dziennego.

**Tabela 3.9:** Usługa zewnętrzna: Where the ISS at?

## 3.5 Analiza ryzyka

# Rozdział 4

## Analiza wymagań

### 4.1 Przypadki użycia

#### 4.1.1 Aktorzy

Niniejszy rozdział przedstawia aktorów wraz z opisami.

**Użytkownik systemu** - Reprezentuje każdą osobę korzystającą z aplikacji.

**Użytkownik niezalogowany** - Gość przeglądający publiczne treści (mapa, spoty, forum): może się zarejestrować lub zalogować.

**Użytkownik zalogowany** - Ma dostęp do wszystkich darmowych funkcjonalności aplikacji. Zarządza kontem i ulubionymi spotami, dodaje posty i komentarze, korzysta z czatu.

**Użytkownik premium** - Użytkownik z wykupioną subskrypcją: ma dostęp do funkcji premium np. oznaczenie stref [PANSÁ](#) na mapie.

**System Finansowo-księgowy** - zewnętrzny system do prowadzenia księgowości, wystawiania faktur oraz rozliczania płatności.

**Usługa SMTP** - usługa Simple Mail Transfer Protocol wykorzystywana do wysyłania wiadomości e-mail.

**Bramka Płatnicza** - usługa obsługująca płatności elektroniczne (karta płatnicza, BLIK itp.).

**Usługa OAuth** - usługa uwierzytelniania i autoryzacji użytkowników z wykorzystaniem zewnętrznych dostawców tożsamości.

**Usługa do przechowywania plików w chmurze** - magazyn plików w chmurze służący do przechowywania załączników i multimediów użytkowników.

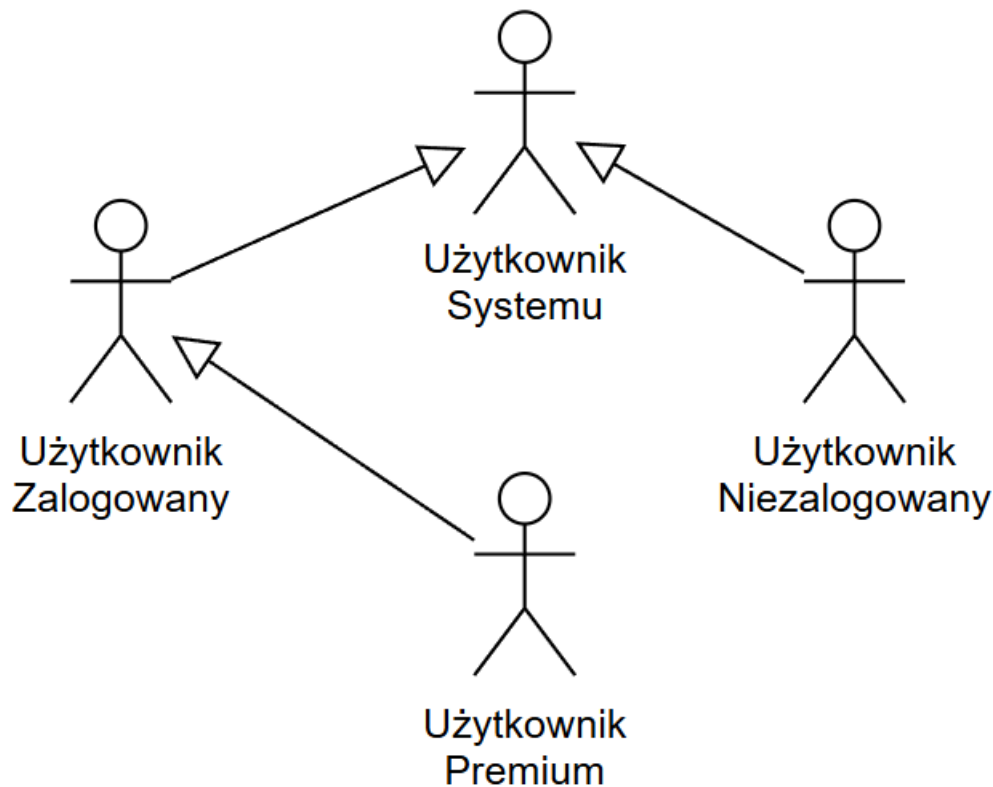
**Usługa do wyświetlania mapy** - zewnętrzne API dostarczające kafelki map, nawigację oraz dane geolokalizacyjne.

**Usługa danych pogodowych** - usługa udostępniająca bieżące warunki pogodowe oraz prognozy dla wybranych lokalizacji.

**Usługa do GIF'ów** - serwis umożliwiający wyszukiwanie i osadzanie animowanych obrazów GIF w aplikacji.

**Usługa do określania strefy czasowej** - usługa ustalająca strefę czasową spota na podstawie jego współrzędnych geograficznych.



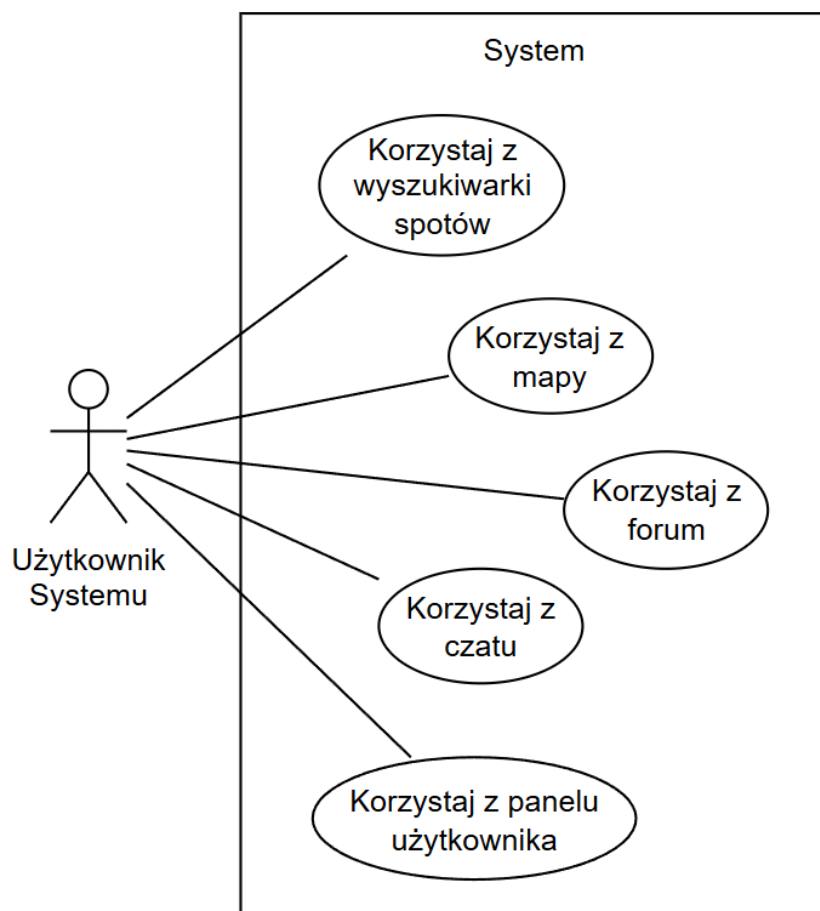


Rysunek 4.1: Diagram hierarchii użytkowników systemu

Na diagramie przedstawiono hierarchię aktorów systemu reprezentujących użytkownika. Podstawową rolą jest Użytkownik systemu, która reprezentuje każdą osobę korzystającą z aplikacji. Z niej dziedziczą dwie bardziej szczegółowe role: Użytkownik niezalogowany (ma dostęp tylko do funkcji publicznych) oraz Użytkownik zalogowany (posiada konto i dostęp do funkcji wymagających uwierzytelnienia). Użytkownik premium jest wyspecjalizowaną wersją użytkownika zalogowanego i oprócz standardowych możliwości ma także dostęp do opcji premium.

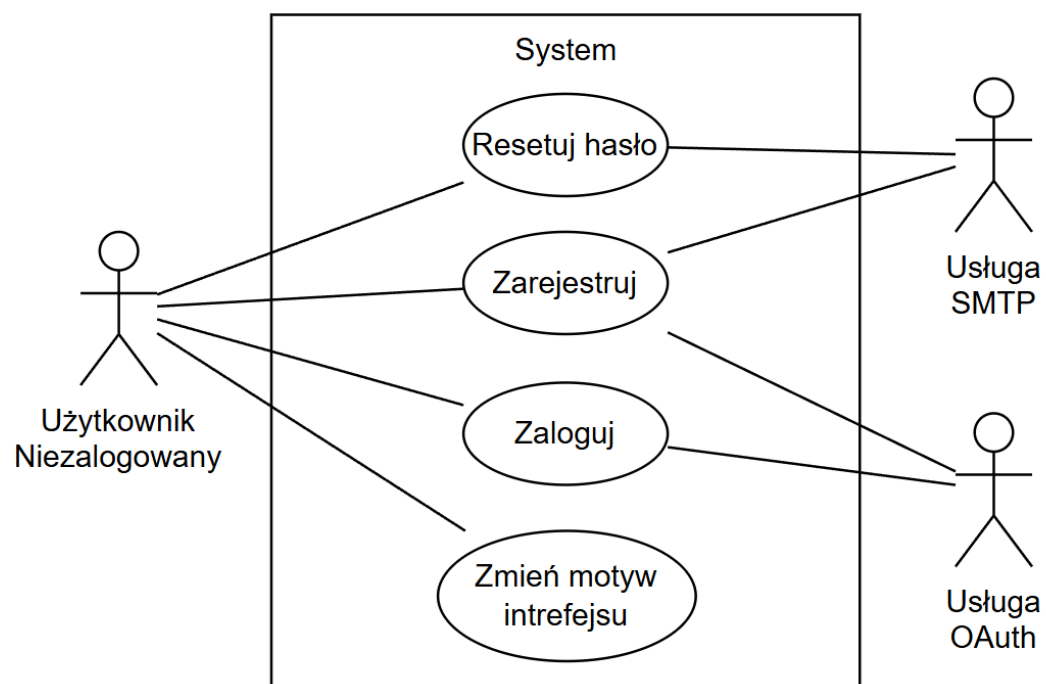
#### 4.1.2 Diagramy przypadków użycia

Niniejszy rozdział przedstawia diagramy przypadków użycia.

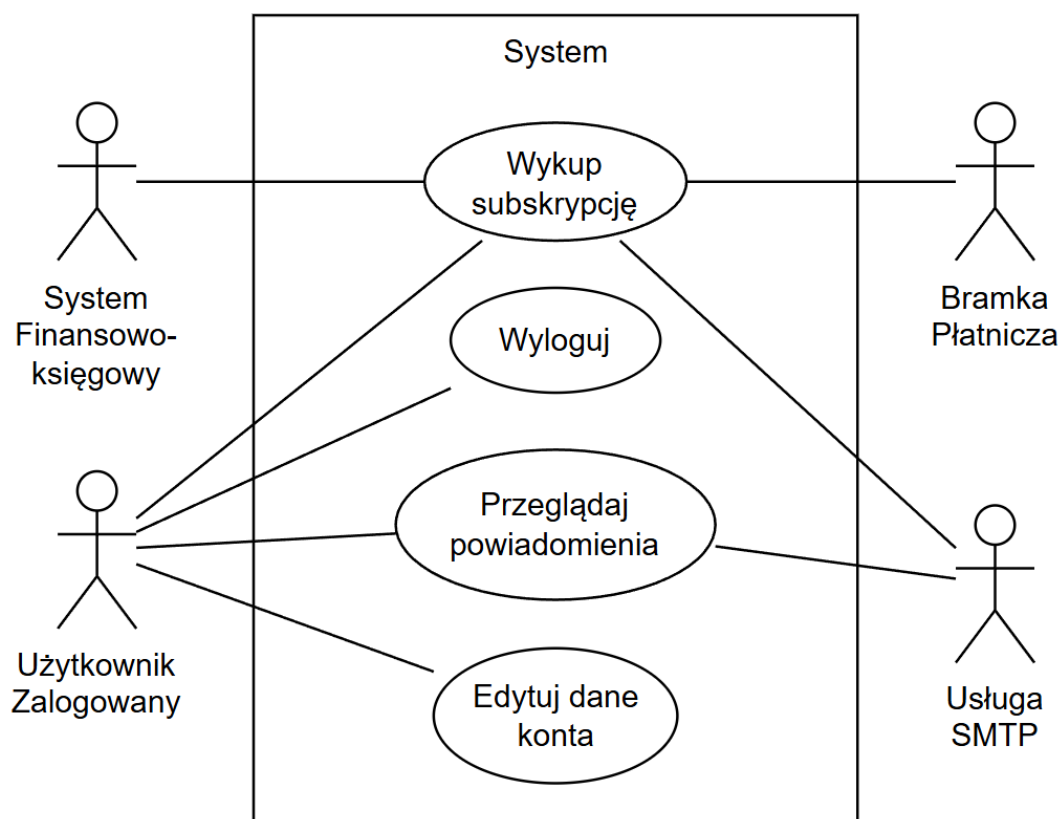


Rysunek 4.2: Wysokopoziomowy diagram przypadków użycia

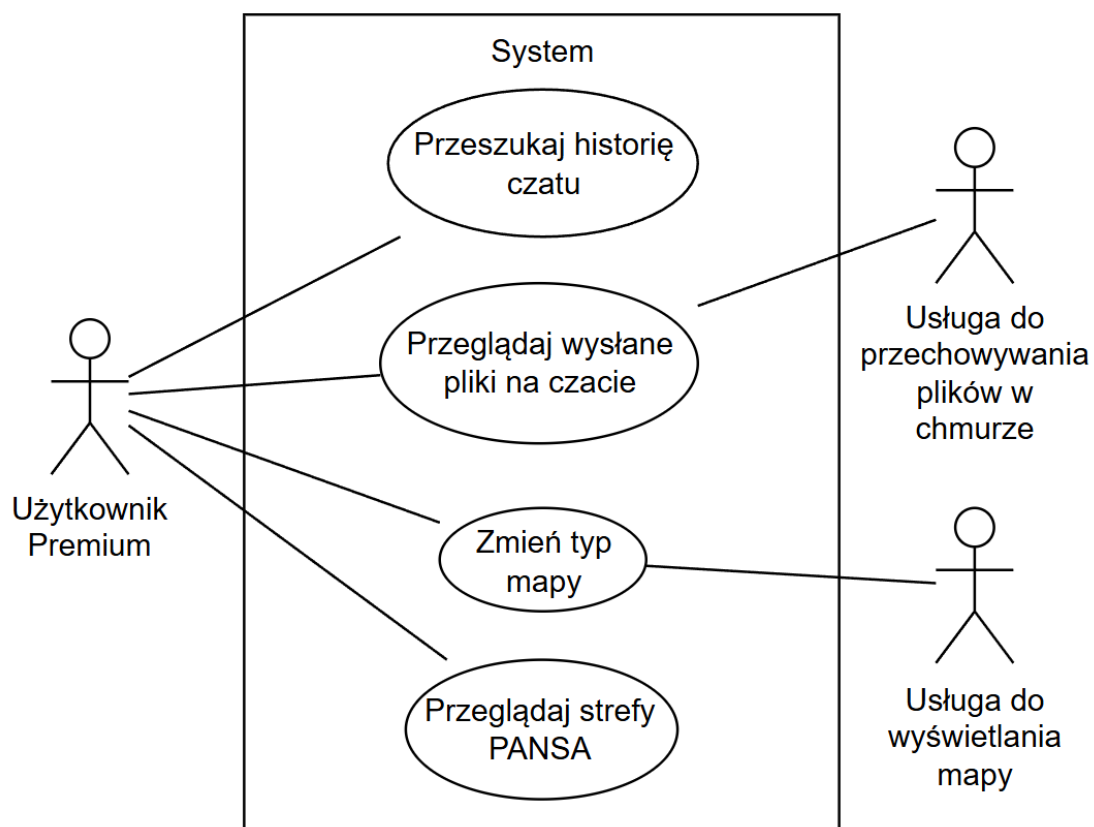
Diagram przedstawia podstawowe interakcje użytkownika z systemem. Na jego podstawie zespół projektowy podzielił architekturę aplikacji na 5 modułów: wyszukiwarkę spotów, mapę spotów, forum, czat oraz profil użytkownika. Pozostałe diagramy są bardziej szczegółowe.



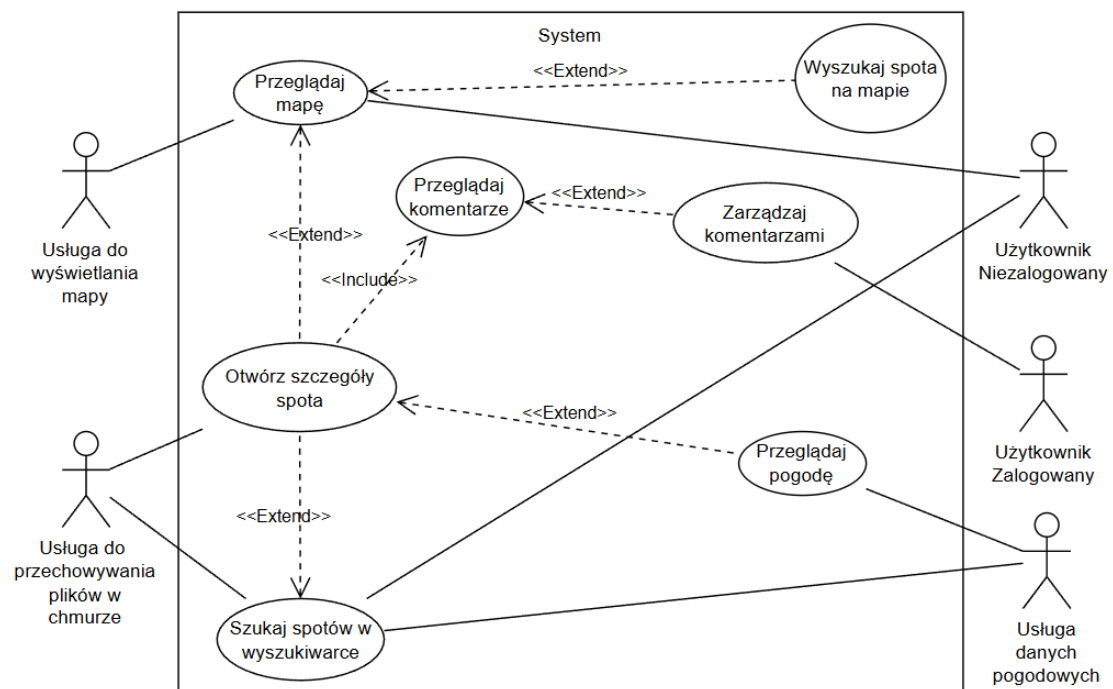
Rysunek 4.3: Diagram przypadków użycia dla użytkownika niezalogowanego



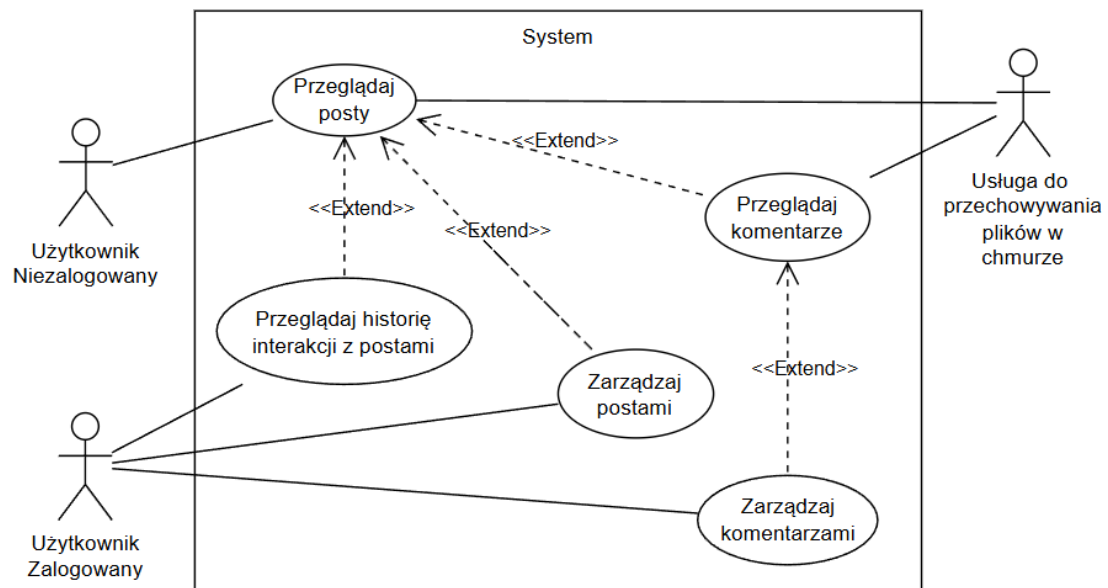
Rysunek 4.4: Diagram przypadków użycia dla użytkownika zalogowanego



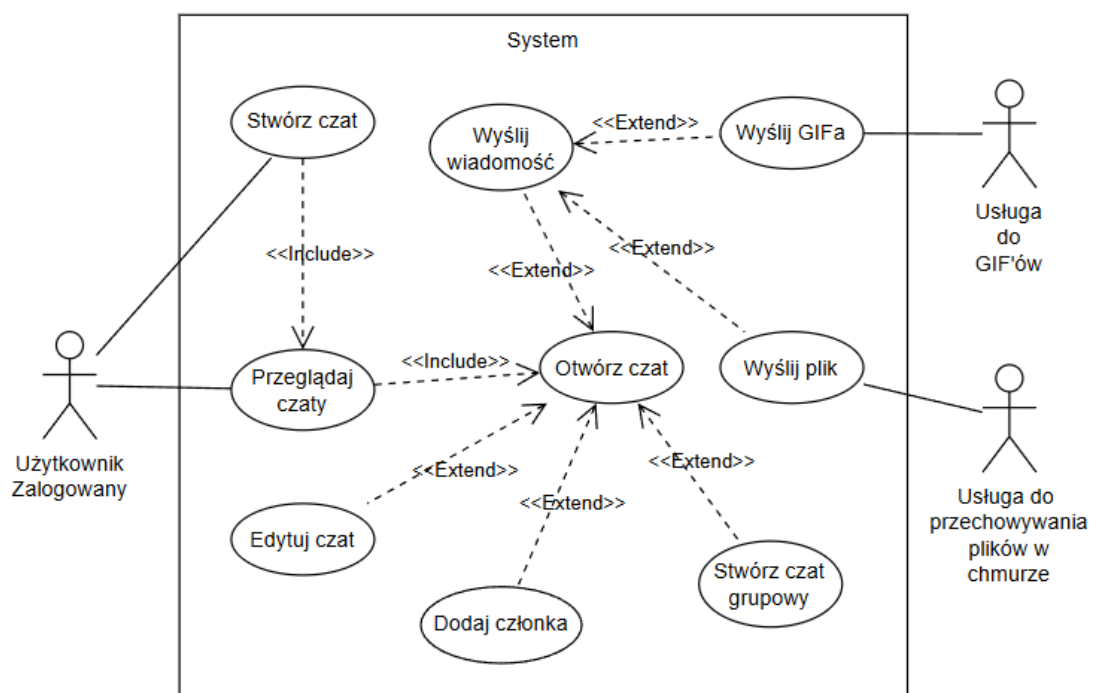
Rysunek 4.5: Diagram przypadków użycia dla użytkownika premium



Rysunek 4.6: Diagram przypadków użycia wyszukiwarki spotów oraz mapy

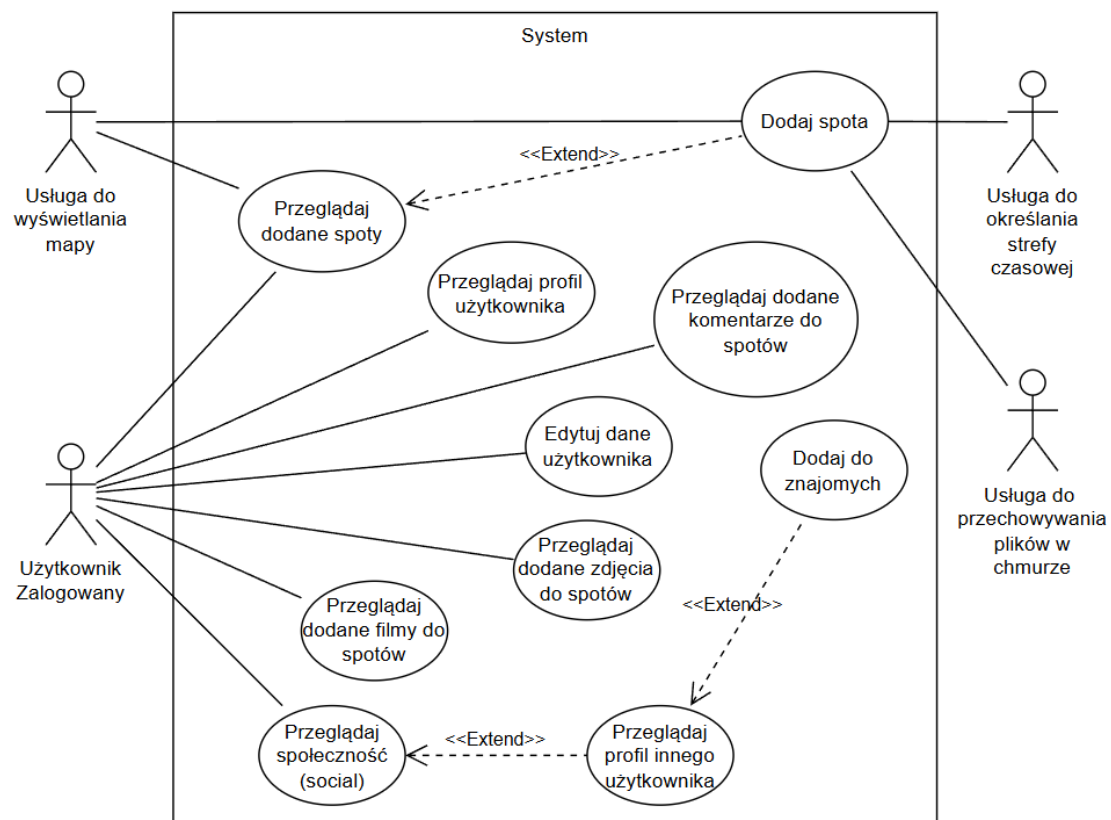


Rysunek 4.7: Diagram przypadków użycia forum



Rysunek 4.8: Diagram przypadków użycia czatu





Rysunek 4.9: Diagram przypadków użycia profilu użytkownika

#### 4.1.3 Scenariusz przypadków użycia

### 4.2 Wymagania ogólne i dziedzinowe

### 4.3 Wymagania funkcjonalne

#### 4.3.1 Funkcjonalności dla mapy

#### 4.3.2 Funkcjonalności dla chatu

#### 4.3.3 Funkcjonalności dla forum

#### 4.3.4 Funkcjonalności dla konta użytkownika

KARTA WYMAGANIA			
Identyfikator:	jednoznaczny symbol np. FO1, FO2 ..	Priorytet:	M
Nazwa:	Profil użytkownika		
Opis:	Jako użytkownik chcę mieć dostęp do strony profilu, aby sprawdzić informacje o swoim koncie.		
Kryteria akceptacji:	Użytkownik widzi liczby: znajomych, obserwowanych i obserwujących, a także najpopularniejsze zdjęcia.		
Dane wejściowe:	Lista zdjęć oraz liczby: znajomych, obserwujących i obserwowanych.		
Warunki początkowe:	Użytkownik jest zalogowany.		
Warunki końcowe:	Wyświetlone informacje o profilu.		
Sytuacje wyjątkowe:	Błąd połączenia z API; brak danych profilu; brak uprawnień (401/403).		
Szczegóły implementacji:	Frontend: React + Tailwind; pobieranie danych profilu przez @tanstack/react-query i axios z withCredentials. Prezentacja w widoku profilu.		
Udziałowiec:	Zespół projektowy 2.1; promotor 2.2; droniarze 2.3.		
Wymagania powiązane:			

Tabela 4.1: Profil użytkownika

KARTA WYMAGANIA			
Identyfikator:	jednoznaczny symbol np. FO1, FO2 ..	Priorytet:	M
Nazwa:	Lista dodanych spotów		
Opis:	Jako użytkownik chcę sprawdzić listę spotów, które dodałem.		
Kryteria akceptacji:	Użytkownik widzi listę własnych dodanych spotów.		
Dane wejściowe:	Lista dodanych spotów.		
Warunki początkowe:	Użytkownik jest zalogowany.		
Warunki końcowe:	Wyświetlona lista dodanych spotów.		
Sytuacje wyjątkowe:	Brak wyników; błąd połączenia z API.		
Szczegóły implementacji:	Pobranie listy z backendu (endpoint listy własnych spotów) przez <code>react-query</code> + <code>axios</code> ; prezentacja listy z podstawowymi danymi.		
Udziałowiec:	Zespół projektowy 2.1; promotor 2.2; droniarze 2.3.		
Wymagania powiązane:			

Tabela 4.2: Lista dodanych spotów

KARTA WYMAGANIA			
Identyfikator:	jednoznaczny symbol np. FO1, FO2 ..	Priorytet:	M
Nazwa:	Dodanie spota		
Opis:	Jako użytkownik chcę mieć dostęp do formularza dodania spota.		
Kryteria akceptacji:	Użytkownik ma dostęp do formularza dodania spota i może go wysłać.		
Dane wejściowe:	Formularz dodania spota.		
Warunki początkowe:	Użytkownik jest zalogowany.		
Warunki końcowe:	Wyświetlony formularz dodania spota (po wysłaniu: zapis na backendzie).		
Sytuacje wyjątkowe:	Nieprawidłowe dane formularza; błąd połączenia z API.		
Szczegóły implementacji:	Formularz w React; walidacja przeglądarkowa; wysyłka przez <code>axios</code> (POST) z <code>withCredentials</code> .		
Udziałowiec:	Zespół projektowy <a href="#">2.1</a> ; promotor <a href="#">2.2</a> ; droniarze <a href="#">2.3</a> .		
Wymagania powiązane:			

Tabela 4.3: Dodanie spota

KARTA WYMAGANIA			
Identyfikator:	jednoznaczny symbol np. FO1, FO2 ..	Priorytet:	M
Nazwa:	Lista zdjęć		
Opis:	Jako użytkownik chcę mieć dostęp do listy zdjęć, które dodałem na forum, do komentarzy pod spotem oraz do spota.		
Kryteria akceptacji:	Użytkownik widzi listę swoich zdjęć.		
Dane wejściowe:	Lista zdjęć.		
Warunki początkowe:	Użytkownik jest zalogowany.		
Warunki końcowe:	Wyświetlona lista zdjęć.		
Sytuacje wyjątkowe:	Brak wyników; błąd połączenia z API.		
Szczegóły implementacji:	Pobranie listy zdjęć użytkownika przez <code>react-query</code> + <code>axios</code> ; prezentacja z miniaturowymi.		
Udziałowiec:	Zespół projektowy 2.1; promotor 2.2; droniarze 2.3.		
Wymagania powiązane:			

Tabela 4.4: Lista zdjęć

KARTA WYMAGANIA			
Identyfikator:	jednoznaczny symbol np. FO1, FO2 ..	Priorytet:	M
Nazwa:	Lista filmów		
Opis:	Jako użytkownik chcę mieć dostęp do listy filmów, które dodałem na forum, do komentarzy pod spotem oraz do spota.		
Kryteria akceptacji:	Użytkownik widzi listę swoich filmów.		
Dane wejściowe:	Lista filmów.		
Warunki początkowe:	Użytkownik jest zalogowany.		
Warunki końcowe:	Wyświetlona lista filmów.		
Sytuacje wyjątkowe:	Brak wyników; błąd połączenia z API.		
Szczegóły implementacji:	Pobranie listy filmów użytkownika przez <code>react-query</code> + <code>axios</code> ; prezentacja z miniaturowymi.		
Udziałowiec:	Zespół projektowy 2.1; promotor 2.2; droniarze 2.3.		
Wymagania powiązane:			

Tabela 4.5: Lista filmów

KARTA WYMAGANIA			
Identyfikator:	jednoznaczny symbol np. FO1, FO2 ..	Priorytet:	M
Nazwa:	Lista znajomych		
Opis:	Jako użytkownik chcę mieć dostęp do listy znajomych.		
Kryteria akceptacji:	Użytkownik ma dostęp do listy znajomych.		
Dane wejściowe:	Lista znajomych.		
Warunki początkowe:	Użytkownik jest zalogowany.		
Warunki końcowe:	Wyświetlona lista znajomych.		
Sytuacje wyjątkowe:	Brak wyników; błąd połączenia z API.		
Szczegóły implementacji:	Pobranie listy znajomych przez <code>react-query</code> + <code>axios</code> ; standardowa prezentacja listy.		
Udziałowiec:	Zespół projektowy 2.1; promotor 2.2; droniarze 2.3.		
Wymagania powiązane:			

Tabela 4.6: Lista znajomych

KARTA WYMAGANIA			
Identyfikator:	jednoznaczny symbol np. FO1, FO2 ..	Priorytet:	M
Nazwa:	Lista obserwujących		
Opis:	Jako użytkownik chcę mieć dostęp do listy obserwujących.		
Kryteria akceptacji:	Użytkownik ma dostęp do listy obserwujących.		
Dane wejściowe:	Lista obserwujących.		
Warunki początkowe:	Użytkownik jest zalogowany.		
Warunki końcowe:	Wyświetlona lista obserwujących.		
Sytuacje wyjątkowe:	Brak wyników; błąd połączenia z API.		
Szczegóły implementacji:	Pobranie listy obserwujących przez <code>react-query</code> + <code>axios</code> ; standardowa prezentacja listy.		
Udziałowiec:	Zespół projektowy 2.1; promotor 2.2; droniarze 2.3.		
Wymagania powiązane:			

Tabela 4.7: Lista obserwujących

KARTA WYMAGANIA			
Identyfikator:	jednoznaczny symbol np. FO1, FO2 ..	Priorytet:	M
Nazwa:	Lista obserwowanych		
Opis:	Jako użytkownik chcę mieć dostęp do listy obserwowanych.		
Kryteria akceptacji:	Użytkownik ma dostęp do listy obserwowanych.		
Dane wejściowe:	Lista obserwowanych.		
Warunki początkowe:	Użytkownik jest zalogowany.		
Warunki końcowe:	Wyświetlona lista obserwowanych.		
Sytuacje wyjątkowe:	Brak wyników; błąd połączenia z API.		
Szczegóły implementacji:	Pobranie listy obserwowanych przez <code>react-query</code> + <code>axios</code> ; standardowa prezentacja listy.		
Udziałowiec:	Zespół projektowy 2.1; promotor 2.2; droniarze 2.3.		
Wymagania powiązane:			

Tabela 4.8: Lista obserwowanych

KARTA WYMAGANIA			
Identyfikator:	jednoznaczny symbol np. FO1, FO2 ..	Priorytet:	M
Nazwa:	Lista spotów		
Opis:	Jako użytkownik chcę mieć dostęp do listy spotów, które polubiłem, odwiedziłem i planuję odwiedzić.		
Kryteria akceptacji:	Użytkownik ma dostęp do listy spotów w wymienionych kategoriach.		
Dane wejściowe:	Listy spotów: polubione, odwiedzone, planowane.		
Warunki początkowe:	Użytkownik jest zalogowany.		
Warunki końcowe:	Wyświetlone listy spotów.		
Sytuacje wyjątkowe:	Brak wyników; błąd połączenia z API.		
Szczegóły implementacji:	Pobranie list przez <code>react-query</code> + <code>axios</code> ; prezentacja w zakładkach/kategoriach.		
Udziałowiec:	Zespół projektowy 2.1; promotor 2.2; droniarze 2.3.		
Wymagania powiązane:			

Tabela 4.9: Lista polubionych/odwiedzonych/planowanych spotów

KARTA WYMAGANIA			
Identyfikator:	jednoznaczny symbol np. FO1, FO2 ..	Priorytet:	M
Nazwa:	Lista komentarzy		
Opis:	Jako użytkownik chcę mieć dostęp do listy komentarzy.		
Kryteria akceptacji:	Użytkownik ma dostęp do listy swoich komentarzy.		
Dane wejściowe:	Lista komentarzy.		
Warunki początkowe:	Użytkownik jest zalogowany.		
Warunki końcowe:	Wyświetlona lista komentarzy.		
Sytuacje wyjątkowe:	Brak wyników; błąd połączenia z API.		
Szczegóły implementacji:	Pobranie listy komentarzy użytkownika przez <code>react-query</code> + <code>axios</code> ; standardowa prezentacja listy.		
Udziałowiec:	Zespół projektowy <a href="#">2.1</a> ; promotor <a href="#">2.2</a> ; droniarze <a href="#">2.3</a> .		
Wymagania powiązane:			

Tabela 4.10: Lista komentarzy



KARTA WYMAGANIA			
Identyfikator:	jednoznaczny symbol np. FO1, FO2 ..	Priorytet:	M
Nazwa:	Ustawienia		
Opis:	Jako użytkownik chcę mieć możliwość zmiany danych.		
Kryteria akceptacji:	Użytkownik może edytować wybrane dane profilu i zapisać zmiany.		
Dane wejściowe:	Formularz edycji danych.		
Warunki początkowe:	Użytkownik jest zalogowany.		
Warunki końcowe:	Wyświetlony formularz edycji; po zapisie — zaktualizowane dane.		
Sytuacje wyjątkowe:	Nieprawidłowe dane formularza; błąd połączenia z API.		
Szczegóły implementacji:	Formularz w React; walidacja pól; wysyłka przez <code>axios</code> (PUT/PATCH) z <code>withCredentials</code> . Po sukcesie — komunikat i odświeżenie danych przez <code>react-query</code> .		
Udziałowiec:	Zespół projektowy <a href="#">2.1</a> ; promotor <a href="#">2.2</a> ; droniarze <a href="#">2.3</a> .		
Wymagania powiązane:			

Tabela 4.11: Ustawienia profilu

KARTA WYMAGANIA			
Identyfikator:	jednoznaczny symbol np. FO1, FO2 ..	Priorytet:	M
Nazwa:	Resetowanie hasła		
Opis:	Jako użytkownik chcę mieć możliwość zresetowania hasła do swojego konta.		
Kryteria akceptacji:	Po kliknięciu w odpowiedni link użytkownik może zresetować hasło do konta.		
Dane wejściowe:	Adres e-mail użytkownika do wysłania linku resetującego.		
Warunki początkowe:	Użytkownik podał poprawny adres e-mail użyty przy rejestracji.		
Warunki końcowe:	Hasło zresetowane po przejściu całej procedury.		
Sytuacje wyjątkowe:	Niepoprawny adres e-mail; wygasły lub nieprawidłowy token resetu; błąd połączenia z API.		
Szczegóły implementacji:	Frontend: formularz „zapomniałem hasła” (POST do endpointu wysyłającego link resetu) oraz formularz ustawienia nowego hasła (POST/PATCH z tokenem). Wysyłka przez <code>axios</code> ; obsługa komunikatów o powodzeniu/błędach.		
Udziałowiec:	Zespół projektowy <a href="#">2.1</a> ; promotor <a href="#">2.2</a> ; droniarze <a href="#">2.3</a> .		
Wymagania powiązane:			

Tabela 4.12: Resetowanie hasła

KARTA WYMAGANIA			
Identyfikator:	jednoznaczny symbol np. FO1, FO2 ..	Priorytet:	M
Nazwa:	Dodawanie użytkowników do listy znajomych		
Opis:	Jako użytkownik chcę mieć możliwość dodawania innych użytkowników do listy znajomych.		
Kryteria akceptacji:	Użytkownik może dodać innego użytkownika do swojej listy znajomych.		
Dane wejściowe:	Dane użytkownika, którego chcemy dodać do znajomych.		
Warunki początkowe:	Użytkownik jest zalogowany.		
Warunki końcowe:	Znajomy dodany do listy i widoczny w profilu użytkownika.		
Sytuacje wyjątkowe:	Brak uprawnień; użytkownik już jest znajomym; błąd połączenia z API.		
Szczegóły implementacji:	Akcja wysłania zaproszenia do znajomych przez <code>axios</code> ; po akceptacji — aktualizacja listy (odświeżenie <code>react-query</code> ).		
Udziałowiec:	Zespół projektowy <a href="#">2.1</a> ; promotor <a href="#">2.2</a> ; droniarze <a href="#">2.3</a> .		
Wymagania powiązane:			

Tabela 4.13: Dodawanie do znajomych

### 4.3.5 Funkcjonalności dla logowania i rejestracji

KARTA WYMAGANIA			
Identyfikator:	FOXX	Priorytet:	M
Nazwa:	Logowanie i rejestracja		
Opis:	Jako użytkownik chcę mieć możliwość zalogowania się do aplikacji, korzystając z formularza lub poprzez konto Google lub GitHub.		
Kryteria akceptacji:	Użytkownik może zalogować się do aplikacji zarówno za pomocą standardowego formularza, jak i przy użyciu konta w serwisie Google lub GitHub.		
Dane wejściowe:	Dane użytkownika: adres e-mail, hasło; przy rejestracji dodatkowo nazwa użytkownika.		
Warunki początkowe:	Użytkownik niezalogowany.		
Warunki końcowe:	Działające formularze rejestracji i logowania oraz możliwość logowania za pomocą konta Google i GitHub.		
Sytuacje wyjątkowe:	Błędne dane logowania; przerwana lub nieudana autoryzacja u dostawcy (Google/GitHub).		
Szczegóły implementacji:	Frontend: formularze w React; wysyłka żądań przez axios z withCredentials. SSO: integracja z Google i GitHub (OAuth 2.0) z przekierowaniem i ustawieniem sesji po stronie backendu (httpOnly cookie). Obsługa statusu 401 zgodnie z mechanizmem wylogowania.		
Udziałowiec:	Zespół projektowy <a href="#">2.1</a> , promotor <a href="#">2.2</a> , droniarze <a href="#">2.3</a> .		
Wymagania powiązane:			

Tabela 4.14: Logowanie i rejestracja

### 4.3.6 Funkcjonalności dla wyszukiwarki spotów

KARTA WYMAGANIA			
Identyfikator:	FOXX	Priorytet:	M
Nazwa:	Strona główna z podstawowymi filtrami		
Opis:	Jako użytkownik chcę mieć dostęp do strony głównej, która wyświetla karuzelę z najpopularniejszymi spotami oraz listę spotów, które można filtrować.		
Kryteria akceptacji:	Użytkownik widzi karuzelę najpopularniejszych miejsc. Karuzela zawiera zdjęcia, nazwę miejsca i miasto. Użytkownik może filtrować miejsca według lokalizacji (kraj, region, miasto).		
Dane wejściowe:	Lokalizacja użytkownika (kraj, region, miasto); dane z bazy spotów.		
Warunki początkowe:	Użytkownik nie musi być zalogowany.		
Warunki końcowe:	Użytkownik widzi popularne miejsca z wybranego miasta (np. Gdańsk) i może przejść do szczegółów danego miejsca.		
Sytuacje wyjątkowe:	Brak wyników dla wybranych filtrów; błąd połączenia z API.		
Szczegóły implementacji:	Frontend: React + Tailwind. Pobieranie danych przez @tanstack/react-query i axios (GET do backendu z parametrami lokalizacji). Filtry lokalizacji mapowane na parametry zapytania.		
Udziałowiec:	Zespół projektowy 2.1, promotor 2.2, droniarze 2.3.		
Wymagania powiązane:			

Tabela 4.15: Strona główna — podstawowe filtry

KARTA WYMAGANIA			
Identyfikator:	FOXX	Priorytet:	M
Nazwa:	Strona główna z zaawansowanymi filtrami		
Opis:	Jako użytkownik chcę mieć dostęp do strony głównej, która wyświetla listę spotów, które można filtrować i sortować.		
Kryteria akceptacji:	Użytkownik widzi listę, którą może filtrować według miasta, tagów i oceny spotu, a także sortować po ocenie i popularności.		
Dane wejściowe:	Lokalizacja użytkownika (miasto), wartości filtrów i sortowania; dane z bazy spotów.		
Warunki początkowe:	Użytkownik nie musi być zalogowany.		
Warunki końcowe:	Użytkownik widzi wyniki zgodne z zastosowanymi filtrami i sortowaniem oraz może przejść do szczegółów danego miejsca.		
Sytuacje wyjątkowe:	Brak wyników po zastosowaniu filtrów; błąd połączenia z API.		
Szczegóły implementacji:	Frontend: React + Tailwind. Pobieranie danych przez <code>@tanstack/react-query</code> i <code>axios</code> z parametrami: lokalizacja, tagi, minimalna ocena oraz kryterium sortowania.		
Udziałowiec:	Zespół projektowy 2.1, promotor 2.2, droniarze 2.3.		
Wymagania powiązane:	SPXX		

Tabela 4.16: Strona główna — zaawansowane filtry

### 4.3.7 Funkcjonalności dla motywu

KARTA WYMAGANIA			
Identyfikator:	FOXX	Priorytet:	M
Nazwa:	Ustawienia motywu		
Opis:	Jako użytkownik chcę móc zmienić motyw aplikacji.		
Kryteria akceptacji:	Dostępna jest opcja przełączenia motywu na <i>jaśny</i> lub <i>ciemny</i> ; zmiana następuje bez przeładowania strony; ustawienie działa we wszystkich widokach.		
Dane wejściowe:	Preferencje użytkownika dotyczące motywu.		
Warunki początkowe:	Brak.		
Warunki końcowe:	Zmiana motywu widoczna jest natychmiast po kliknięciu przycisku.		
Sytuacje wyjątkowe:	Brak.		
Szczegóły implementacji:	Tailwind CSS z <code>darkMode: 'class'</code> ; motyw przełączany przez dodanie/usunięcie klasy <code>dark</code> na elemencie <code>&lt;html&gt;</code> ;		
Udziałowiec:	Zespół projektowy 2.1, promotor 2.2, droniarze 2.3.		
Wymagania powiązane:			

Tabela 4.17: Ustawienia motywu (ręczna zmiana)

KARTA WYMAGANIA			
Identyfikator:	FOXX	Priorytet:	M
Nazwa:	Zapamiętywanie preferencji motywu		
Opis:	Jako użytkownik chcę, aby moja preferencja motywu była zapamiętana i przywracana przy kolejnym użyciu aplikacji.		
Kryteria akceptacji:	Wybrany motyw jest przywracany po ponownym włączeniu i odświeżeniu strony; preferencja jest zapamiętywana lokalnie w przeglądarce.		
Dane wejściowe:	Preferencje użytkownika zapisane lokalnie.		
Warunki początkowe:	FOXX dostępne.		
Warunki końcowe:	Motyw po uruchomieniu odpowiada ostatniej decyzji użytkownika.		
Sytuacje wyjątkowe:	Brak dostępu do magazynu trwałego — preferencja przechowywana w local storage.		
Szczegóły implementacji:	Zapis w <code>localStorage</code> pod kluczem <code>theme</code> ( <code>dark</code> lub <code>light</code> ); krótki skrypt umieszczony w <code>App.jsx</code> przed startem odczytuje <code>localStorage</code> i odpowiednio dodaje lub usuwa klasę <code>dark</code> na <code>&lt;html&gt;</code> (eliminuje mignięcie stylów).		
Udziałowiec:	Zespół projektowy 2.1, promotor 2.2, droniarze 2.3.		
Wymagania powiązane:			

Tabela 4.18: Zapamiętywanie preferencji motywu



KARTA WYMAGANIA			
Identyfikator:	FOXX	Priorytet:	S
Nazwa:	Przełącznik motywu w <a href="#">Sidebar</a>		
Opis:	Jako użytkownik chcę szybko zmieniać motyw bez wchodzenia w ustawienia.		
Kryteria akceptacji:	W <a href="#">Sidebar</a> dostępny jest przełącznik <i>Jasny-/Ciemny</i> ; posiada odpowiednio ikony <i>słońca/księżyca</i> ; zmiana następuje natychmiast.		
Dane wejściowe:	Bieżąca preferencja motywu.		
Warunki początkowe:	FOXX, FOXX dostępne.		
Warunki końcowe:	Motyw zmieniony; preferencja zaktualizowana.		
Sytuacje wyjątkowe:	Brak.		
Szczegóły implementacji:	Przycisk typu <i>toggle</i> wywołuje funkcję, która przełącza klasę <code>dark</code> na <code>document.documentElement</code> oraz aktualizuje <code>localStorage</code> ( <code>theme = 'dark' 'light'</code> ); brak przeładowania strony.		
Udziałowiec:	Zespół projektowy <a href="#">2.1</a> , promotor <a href="#">2.2</a> , droniarze <a href="#">2.3</a> .		
Wymagania powiązane:			

Tabela 4.19: Szybki przełącznik motywu w interfejsie

## 4.4 Wymagania pozafunkcjonalne

## 4.5 Wymagania interfejs z otoczeniem

## 4.6 Wymagania na środowisko docelowe

# Rozdział 5

## Projekt

### 5.1 Wzorce projektowe

### 5.2 Architektura systemu

W niniejszym rozdziale przedstawiona zostanie architektura systemu, czyli sposób, w jaki poszczególne komponenty komunikują się między sobą, a także technologie, za pomocą których zostały stworzone.

Jednym z kluczowych etapów realizacji projektu był wybór odpowiedniej architektury systemowej. Ostatecznie przyjęto oddzielenie poszczególnych warstw aplikacji, co zapewnia większą elastyczność, skalowalność oraz ułatwia rozwój w przyszłości. Przyjęte komponenty prezentują się następująco:

- [frontend](#) – [React](#) z wykorzystaniem [TypeScriptu](#),
- [backend](#) – Java Spring Boot,
- [baza danych](#) – PostgreSQL,
- [redis](#) – wykorzystywany jako [baza danych](#) klucz–wartość pełniący rolę warstwy [cache](#).

Jest to podejście, w którym zespół projektowy posiada największe doświadczenie, dlatego zostało ono zastosowane. Pozwala ono również na tworzenie aplikacji responsywnej, dostępnej zarówno na komputerach, jak i urządzeniach mobilnych.

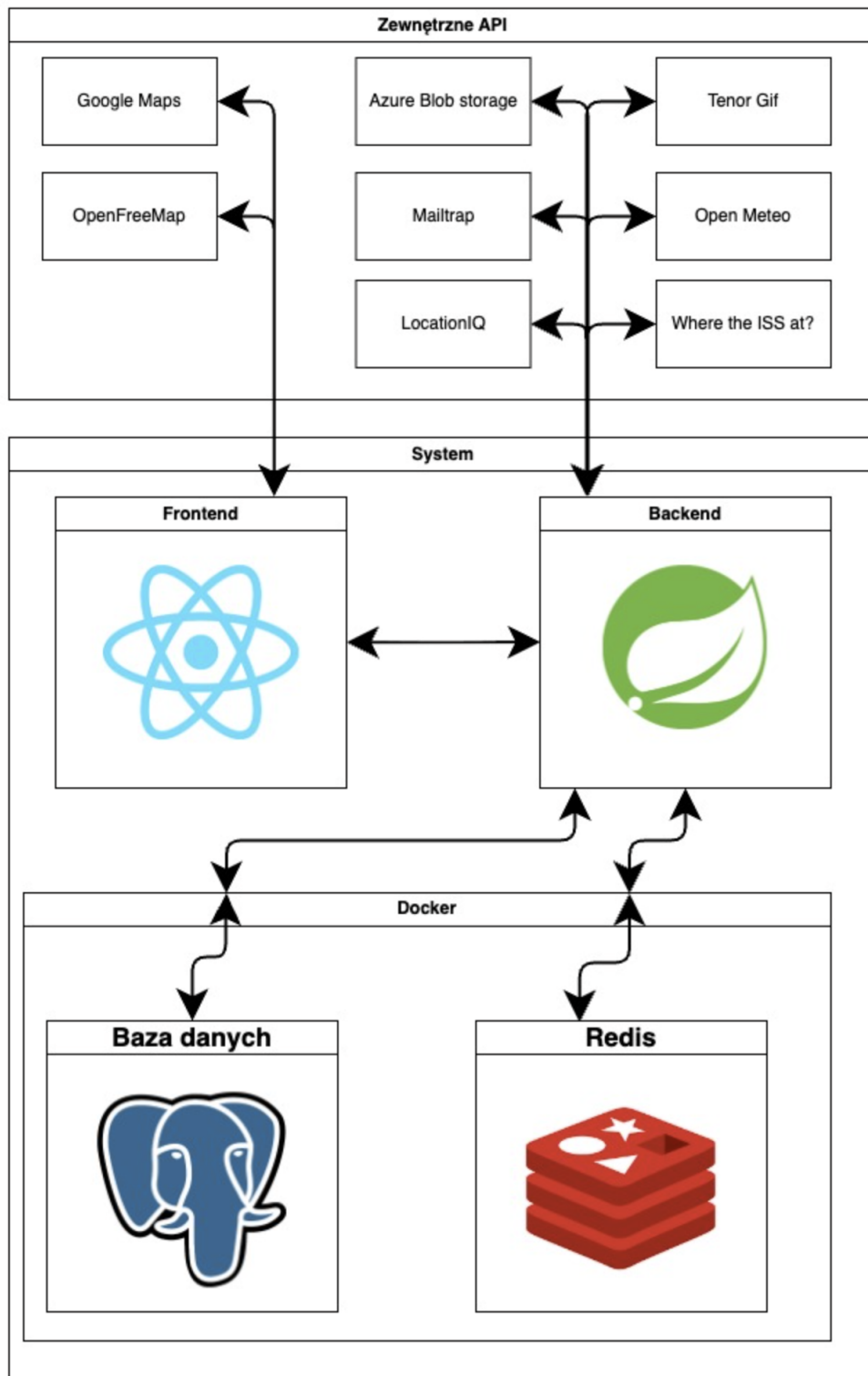
Warstwa wizualna została przygotowana przy użyciu [React](#) w wersji z [TypeScriptem](#) oraz [biblioteki](#) Tailwind CSS, zapewniającej szybkie i wygodne stylowanie komponentów. Z kolei za komunikację oraz logikę biznesową odpowiada [backend](#) oparty na [frameworku](#) Spring Boot, realizujący założenia architektury [REST API](#). Jako system zarządzania danymi wybrano relacyjną bazę danych PostgreSQL, z którą zespół posiada największe doświadczenie. Relacyjny model danych doskonale sprawdza się w tym projekcie, zapewniając integralność danych, możliwość tworzenia złożonych zapytań oraz wysoką stabilność.

[Redis](#) został wykorzystany jako warstwa [cache](#), której zadaniem jest przyspieszenie działania aplikacji poprzez ograniczenie liczby odwołań do głównej [bazy danych](#). Dzięki przechowywaniu często wykorzystywanych danych w pamięci operacyjnej znacznie skraca się czas odpowiedzi systemu, co pozytywnie wpływa na wydajność oraz skalowalność rozwiązania. Zastosowanie [Redisa](#) okazało się szczególnie korzystne w przypadku operacji powtarzalnych i odczytowych, które nie wymagają każdorazowego dostępu do relacyjnej [bazy danych](#).

### 5.2.1 Diagram architektury

Projekt aplikacji oparto na architekturze klient-serwer z podziałem na [frontend](#) i [backend](#). Takie podejście ułatwia rozwój i utrzymanie systemu oraz umożliwia skalowanie poszczególnych komponentów niezależnie od siebie. Komunikacja między [frontendem](#) a [backendem](#) odbywa się za pomocą [REST API](#), przy czym dane przesyłane są w formacie JSON. Integracja między warstwami aplikacji jest dzięki temu lekka, czytelna i łatwa do rozszerzenia w przyszłości.

Architektura została opracowana dla środowiska deweloperskiego. W obecnym zakresie prac nie uwzględniono implementacji środowiska produkcyjnego.



Rysunek 5.1: Diagram architektury

## 5.2.2 Komponenty systemu

System składa się z kilku głównych komponentów, z których każdy pełni ściśle określoną rolę.

- [Frontend](#) – odpowiada za warstwę prezentacji oraz interfejs użytkownika dostępny dla wszystkich użytkowników systemu,
- [Backend](#) – odpowiada za autoryzację użytkowników oraz obsługę komunikacji między [frontendem](#) a [bazą danych](#),
- [Baza danych](#) – przechowuje wszystkie dane aplikacji, w tym dane użytkowników, dane operacyjne oraz informacje potrzebne do działania systemu.
- [Redis](#) – wykorzystywany jako warstwa cache, przechowująca często odczytywane dane w pamięci operacyjnej, co znacząco przyspiesza działanie systemu oraz zmniejsza obciążenie głównej bazy danych.

Szczegółowy wykaz wykorzystywanych zewnętrznych API zamieszczono w rozdziale 3.

- Azure Blob Storage – [3.2](#)
- Mailtrap – [3.3](#)
- LocationIQ – [3.4](#)
- Google Maps – [3.5](#)
- OpenFreeMap – [3.6](#)
- Open Meteo – [3.7](#)
- Tenor Gif – [3.8](#)
- Where the ISS at? – [3.9](#)

## 5.3 Projekt bazy danych

### 5.3.1 Model danych

### 5.3.2 Diagram ERD

## 5.4 Architektura interfejsu użytkownika

### 5.4.1 Projekt strony głównej

### 5.4.2 Projekt panelu logowania

### 5.4.3 Projekt mapy

### 5.4.4 Projekt chatu

### 5.4.5 Projekt forum

### 5.4.6 Projekt konta użytkownika

## Rozdział 6

# Przebieg realizacji projektu

### 6.1 Sprint 1

### 6.2 Sprint 2

# Rozdział 7

## Realizacja Projektu

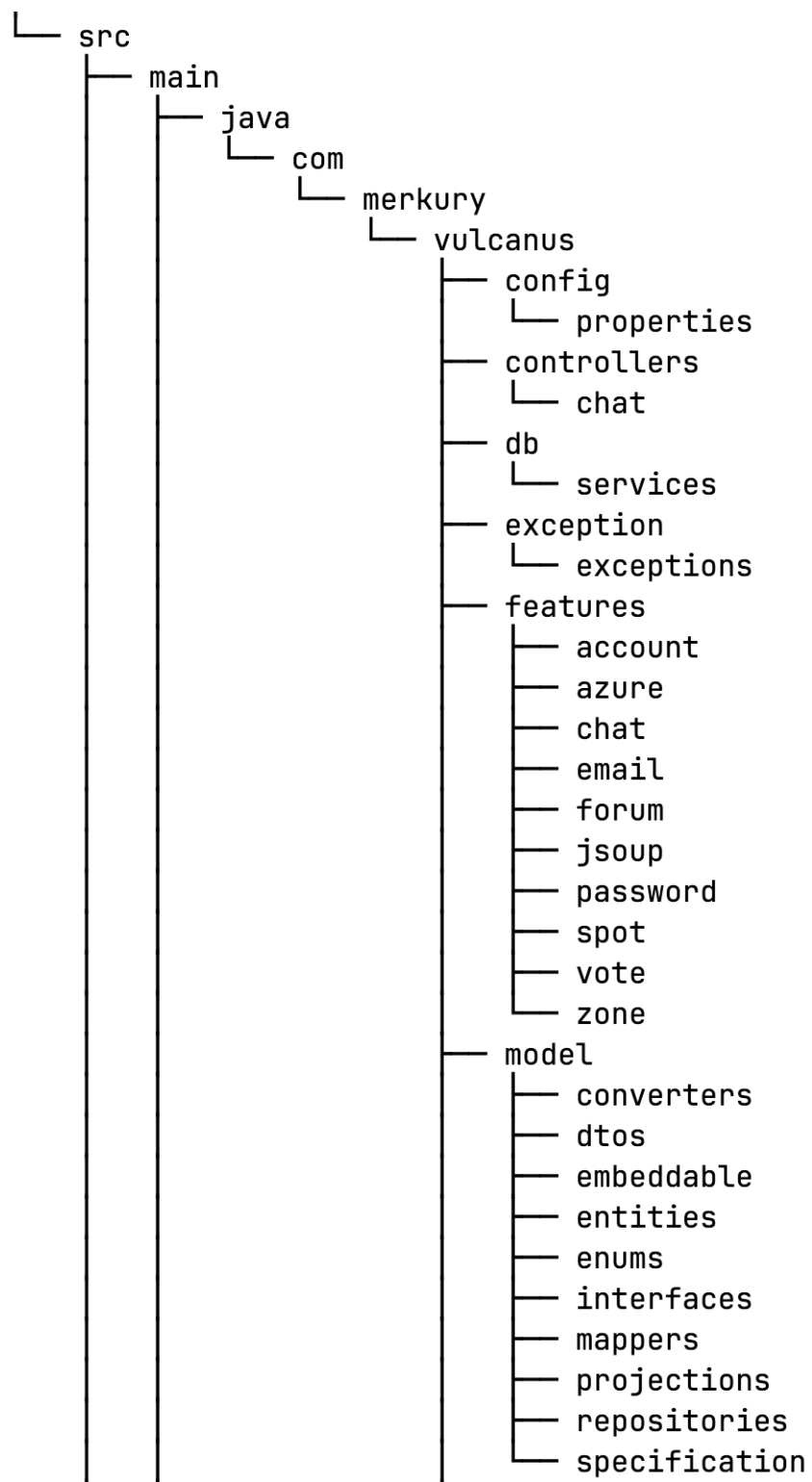
### 7.1 Implementacja backendu

W niniejszym rozdziale przedstawiono strukturę backendu aplikacji, opis implementowanych endpointów, integrację z bazą danych, mechanizmy uwierzytelniania oraz proces konteneryzacji systemu.

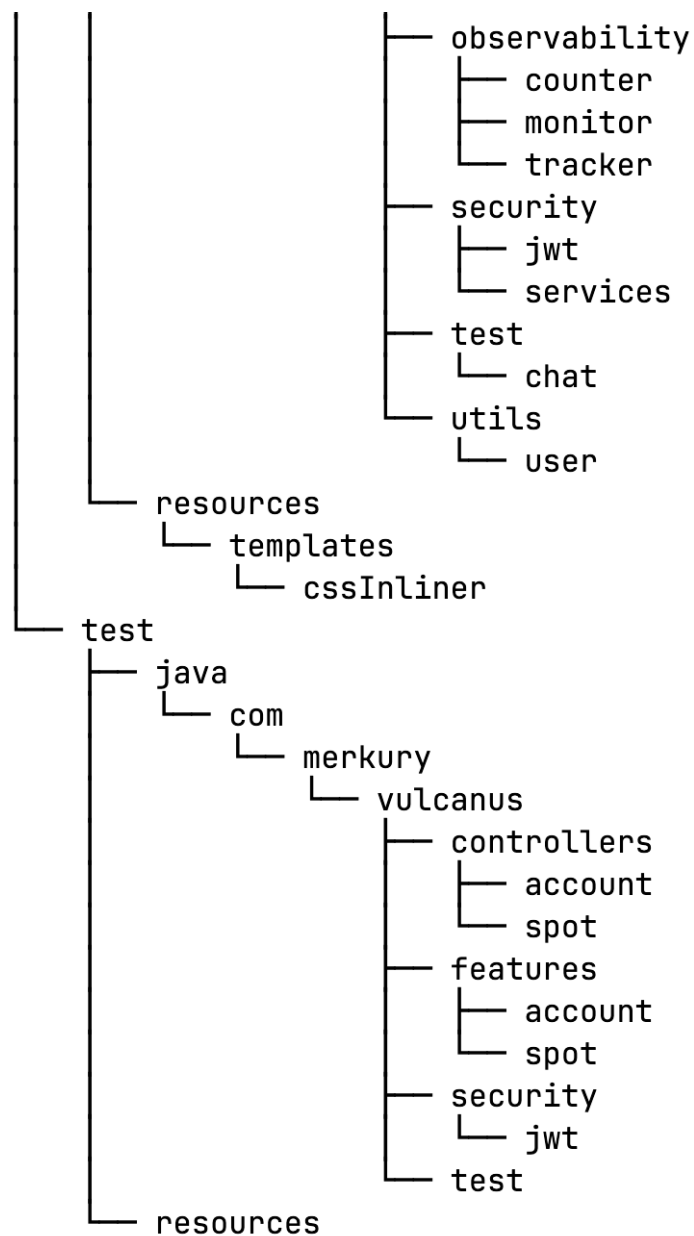
#### 7.1.1 Struktura projektu

[Backend](#) aplikacji został zaimplementowany przy użyciu [frameworka](#) Spring Boot, co umożliwiło stworzenie spójnej i skalowalnej architektury w prosty sposób. W projekcie zastosowano rozwiązanie typu [REST API](#), gdyż zespół projektowy dysponuje największym doświadczeniem w jego wykorzystaniu. Struktura projektu została zorganizowana zgodnie z podejściem [folder by type](#), dzięki czemu każdy plik znajduje się w odpowiadającym mu katalogu. Takie podejście ułatwia zarówno lokalizację istniejących plików, jak i określenie miejsca tworzenia nowych komponentów. Poniżej przedstawiono przykładową strukturę katalogów [backendu](#):





Rysunek 7.1: Struktura katalogów (1)



Rysunek 7.2: Struktura katalogów (2)

Dzięki takiej organizacji kod jest bardziej czytelny i łatwiejszy w utrzymaniu. Umożliwia również szybkie odnalezienie odpowiednich modułów oraz ułatwia rozbudowę projektu w przyszłości.

### 7.1.2 Endpointy systemu

Projektowany system udostępnia [REST-owe API HTTP](#), za pomocą którego klienci komunikują się z serwerem.

Na potrzeby niniejszej pracy przez *endpoint* [REST API](#) rozumiany będzie konkretny punkt dostępu do systemu, zdefiniowany jako para:

metoda HTTP + ścieżka URL

pod którym aplikacja udostępnia określoną funkcjonalność lub zasób. Przykładowo, endpoint `GET /public/spot/{spotId}` służy do pobierania informacji o wybranym spocie.

W dalszej części rozdziału przedstawiono zestawienie wszystkich endpointów HTTP systemu, a następnie szczegółowe karty wybranych endpointów, opisujące m.in. parametry wejściowe, [Query params](#) oraz strukturę odpowiedzi.

Zestawienie wszystkich endpointów HTTP systemu	
Metoda HTTP	Ścieżka
Panel użytkownika	
GET	/user-dashboard/profile
GET	/public/user-dashboard/profile/{targetUsername}
PATCH	/user-dashboard/profile
GET	/user-dashboard/friends
GET	/public/user-dashboard/friends/{targetUsername}
PATCH	/user-dashboard/friends
PATCH	/user-dashboard/friends/change-status
GET	/user-dashboard/followers
GET	/public/user-dashboard/followers/{targetUsername}
GET	/user-dashboard/followed
GET	/public/user-dashboard/followed/{targetUsername}
GET	/user-dashboard/friends/find

Metoda HTTP	Ścieżka
GET	/user-dashboard/friends/invites
PATCH	/user-dashboard/followed
GET	/user-dashboard/favorite-spots
PATCH	/user-dashboard/favorite-spots
POST	/user-dashboard/add-spot-media
GET	/user-dashboard/is-spot-favourite
GET	/user-dashboard/photos
GET	/user-dashboard/comments
PATCH	/user-dashboard/settings
GET	/user-dashboard/settings
GET	/user-dashboard/movies
GET	/user-dashboard/photos/{targetUsername}
GET	/user-dashboard/add-spot
POST	/user-dashboard/add-spot
GET	/user-dashboard/add-spot/coordinates
<b>Spoty</b>	
GET	/public/spot/gallery
GET	/public/spot/gallery-media-position
GET	/public/spot/gallery-fullscreen-media
GET	/public/spot/current-view
GET	/public/spot/current-view/spot-names
GET	/public/spot/{spotId}
PATCH	/public/spot/increase-view-count
GET	/public/spot/search/map
GET	/public/spot/search/list
GET	/public/spot/names
GET	/public/spot/most-popular
GET	/public/spot/search/home-page

Metoda HTTP	Ścieżka
GET	/public/spot/search/home-page/locations
GET	/public/spot/search/home-page/advance
GET	/public/spot/get-spot-basic-weather
GET	/public/spot/get-spot-detailed-weather
GET	/public/spot/get-spot-wind-speeds
GET	/public/spot/get-spot-weather-timeline-plot-data
PATCH	/public/spot/increase-spot-media-views-count
PATCH	/public/spot/edit-spot-media-likes
GET	/spot/check-is-spot-media-liked
GET	/public/spot/get-spot-time-zone
<b>Komentarze do spotów</b>	
GET	/public/spot/{spotId}/comments
GET	/public/spot/{spotId}/comments/{commentId}
POST	/spot/{spotId}/comments
DELETE	/spot/comments/{commentId}
PATCH	/spot/comments/{commentId}
PATCH	/spot/comments/{commentId}/vote
GET	/spot/comments/vote-type
<b>Posty forum</b>	
GET	/public/post/{postId}
GET	/public/post
POST	/post
DELETE	/post/{postId}
PATCH	/post/{postId}
PATCH	/post/{postId}/vote
PATCH	/post/{postId}/follow
PATCH	/post/{postId}/report
GET	/public/categories-tags

Metoda HTTP	Ścieżka
<b>Komentarze forum</b>	
GET	/public/post/{postId}/comments
GET	/public/comments/{commentId}/replies
POST	/post/{postId}/comments
DELETE	/post/comments/{commentId}
PATCH	/post/comments/{commentId}
PATCH	/post/comments/{commentId}/vote
PATCH	/post/comments/{commentId}/report
POST	/comments/{commentId}/replies
<b>Konto użytkownika / autoryzacja</b>	
POST	/public/account/register
POST	/public/account/login
GET	/account/login-success
POST	/public/account/forgot-password
POST	/public/account/set-new-password
GET	/account/check
<b>Przesyłanie plików</b>	
POST	/public/upload/media
<b>GIF-y (Tenor)</b>	
GET	/gifs/trending
GET	/gifs/search
<b>Czat</b>	
GET	/chats/{chatId}/messages
GET	/chats/user-chats
POST	/chats/get-or-create-private-chat
POST	/chats/{chatId}/send-files
POST	/chats/create/group
PATCH	/chats/{chatId}

Metoda HTTP	Ścieżka
GET	/chats/group-chat/add/search/{chatId}
PUT	/chats/add/users/{chatId}

**Tabela 7.1:** Zestawienie wszystkich endpointów HTTP systemu

#### Panel użytkownika

KARTA ENDPOINTU API	
<b>Identyfikator:</b>	EP01
<b>Ścieżka:</b>	/public/user-dashboard/profile/{targetUsername}
<b>Nazwa:</b>	Pobierz profil innego użytkownika (widok publiczny)
<b>Parametry wejściowe:</b>	<ul style="list-style-type: none"> <li>• <b>targetUsername:</b> String (nazwa użytkownika w ścieżce URL)</li> </ul>
<b>Query params:</b>	Brak
<b>Kod(y) statusu odpowiedzi:</b>	200 OK, 404 Not Found

<b>Dane zwracane:</b>	<ul style="list-style-type: none"> <li>• <b>profile:</b> UserProfileDto, zawiera: <ul style="list-style-type: none"> <li>– <b>username:</b> String</li> <li>– <b>profilePhoto:</b> String (URL)</li> <li>– <b>followersCount:</b> Integer</li> <li>– <b>followedCount:</b> Integer</li> <li>– <b>friendsCount:</b> Integer</li> <li>– <b>photosCount:</b> Integer</li> <li>– <b>mostPopularPhotos:</b> List&lt;ImageDto&gt;</li> </ul> </li> <li>• <b>friendStatus:</b> UserFriendStatus</li> <li>• <b>isFollowing:</b> Boolean</li> <li>• <b>isOwnProfile:</b> Boolean</li> </ul>
---------------------------	--

**Tabela 7.2:** Karta endpointu: /public/user-dashboard/profile/{targetUsername}

KARTA ENDPOINTU API	
<b>Identyfikator:</b>	EP02
<b>Ścieżka:</b>	/user-dashboard/favorite-spots
<b>Nazwa:</b>	Pobierz listę ulubionych spotów użytkownika
<b>Parametry wejściowe:</b>	Brak
<b>Query params:</b>	<ul style="list-style-type: none"> <li>• <b>type:</b> FavoriteSpotsListType (typ listy: ulubione, odwiedzone oraz do odwiedzenia)</li> <li>• <b>page:</b> Integer (opcjonalnie, domyślnie 0)</li> <li>• <b>size:</b> Integer (opcjonalnie, domyślnie 10)</li> </ul>



<b>Kod(y) statusu odpowiedzi:</b>	200 OK, 401 Unauthorized
<b>Dane zwracane:</b>	<ul style="list-style-type: none"> <li>• <b>items:</b> List&lt;FavoriteSpotDto&gt;, gdzie każdy element zawiera: <ul style="list-style-type: none"> <li>– <b>id:</b> Long</li> <li>– <b>name:</b> String</li> <li>– <b>country:</b> String</li> <li>– <b>city:</b> String</li> <li>– <b>description:</b> String</li> <li>– <b>rating:</b> Double</li> <li>– <b>viewsCount:</b> Integer</li> <li>– <b>imageUrl:</b> String</li> <li>– <b>type:</b> FavoriteSpotsListType</li> <li>– <b>coords:</b> SpotCoordinatesDto</li> <li>– <b>tags:</b> Set&lt;SpotTagDto&gt;</li> </ul> </li> <li>• <b>hasNext:</b> boolean</li> </ul>

**Tabela 7.3:** Karta endpointu: /user-dashboard/favorite-spots

KARTA ENDPOINTU API	
<b>Identyfikator:</b>	EP03
<b>Ścieżka:</b>	/user-dashboard/photos
<b>Nazwa:</b>	Pobierz posortowane zdjęcia użytkownika

<b>Parametry wejściowe:</b>	Brak
<b>Query params:</b>	<ul style="list-style-type: none"> <li>• <b>type:</b> DateSortType (typ sortowania po dacie)</li> <li>• <b>from:</b> LocalDate (opcjonalnie, data od)</li> <li>• <b>to:</b> LocalDate (opcjonalnie, data do)</li> <li>• <b>page:</b> Integer (opcjonalnie, domyślnie 0)</li> <li>• <b>size:</b> Integer (opcjonalnie, domyślnie 20)</li> </ul>
<b>Kod(y) statusu odpowiedzi:</b>	200 OK, 400 Bad Request, 401 Unauthorized
<b>Dane zwracane:</b>	<ul style="list-style-type: none"> <li>• <b>items:</b> List&lt;DatedMediaGroupDto&gt;, gdzie: <ul style="list-style-type: none"> <li>– <b>date:</b> LocalDate (data grupy)</li> <li>– <b>media:</b> List&lt;MediaDto&gt;, każdy element: <ul style="list-style-type: none"> <li>* <b>src:</b> String (URL)</li> <li>* <b>heartsCount:</b> Integer</li> <li>* <b>viewsCount:</b> Integer</li> <li>* <b>id:</b> Long</li> </ul> </li> </ul> </li> <li>• <b>hasNext:</b> boolean</li> </ul>

**Tabela 7.4:** Karta endpointu: /user-dashboard/photos

KARTA ENDPOINTU API	
<b>Identyfikator:</b>	EP04
<b>Ścieżka:</b>	/user-dashboard/add-spot
<b>Nazwa:</b>	Pobierz listę spotów dodanych przez użytkownika
<b>Parametry wejściowe:</b>	Brak

<b>Query params:</b>	<ul style="list-style-type: none"> <li>• <b>page:</b> Integer (opcjonalnie, domyślnie 0)</li> <li>• <b>size:</b> Integer (opcjonalnie, domyślnie 20)</li> </ul>
<b>Kod(y) statusu odpowiedzi:</b>	200 OK, 401 Unauthorized
<b>Dane zwracane:</b>	<ul style="list-style-type: none"> <li>• <b>items:</b> List&lt;AddSpotDto&gt;, każdy element: <ul style="list-style-type: none"> <li>– <b>id:</b> Long</li> <li>– <b>name:</b> String</li> <li>– <b>description:</b> String</li> <li>– <b>country:</b> String</li> <li>– <b>region:</b> String</li> <li>– <b>city:</b> String</li> <li>– <b>street:</b> String</li> <li>– <b>borderPoints:</b> List&lt;BorderPoint&gt; (x, y)</li> <li>– <b>firstPhotoUrl:</b> String</li> </ul> </li> <li>• <b>hasNext:</b> boolean</li> </ul>

**Tabela 7.5:** Karta endpointu: /user-dashboard/add-spot

KARTA ENDPOINTU API	
<b>Identyfikator:</b>	EP05
<b>Ścieżka:</b>	/user-dashboard/add-spot
<b>Nazwa:</b>	Dodaj nowy spot użytkownika

<b>Parametry wejściowe:</b>	<ul style="list-style-type: none"> <li>• <b>spot:</b> String (część multipart, JSON z danymi nowego spotu)</li> <li>• <b>media:</b> List&lt;MultipartFile&gt; (część multipart, pliki multimedialne spotu)</li> </ul>
<b>Query params:</b>	Brak
<b>Kod(y) statusu odpowiedzi:</b>	200 OK, 404 Not Found, 401 Unauthorized
<b>Dane zwracane:</b>	Brak (pusta odpowiedź)

**Tabela 7.6:** Karta endpointu: /user-dashboard/add-spot

## Spoty i pogoda

KARTA ENDPOINTU API	
<b>Identyfikator:</b>	EP06
<b>Ścieżka:</b>	/public/spot/gallery
<b>Nazwa:</b>	Pobierz stronę galerii mediów dla spotu
<b>Parametry wejściowe:</b>	Brak
<b>Query params:</b>	<ul style="list-style-type: none"> <li>• <b>spotId:</b> Long (identyfikator spotu)</li> <li>• <b>mediaType:</b> String (typ plików, wartość enum GenericMediaType, PHOTO, VIDEO)</li> <li>• <b>sorting:</b> String (kryterium sortowania, po dacie / popularności)</li> <li>• <b>page:</b> Integer (numer strony, domyślnie 0)</li> <li>• <b>size:</b> Integer (rozmiar strony, domyślnie 6)</li> </ul>

Kod(y) statusu odpowiedzi:	200 OK, 404 Not Found
Dane zwracane:	<ul style="list-style-type: none"> <li>• <b>Page&lt;SpotSidebarMediaGalleryDto&gt;</b>: stronicowana lista elementów galerii, gdzie każdy element zawiera: <ul style="list-style-type: none"> <li>– <b>id</b>: Long (ID media)</li> <li>– <b>url</b>: String (URL pliku)</li> <li>– <b>mediaType</b>: GenericMediaType (typ pliku)</li> </ul> </li> </ul>

**Tabela 7.7:** Karta endpointu: /public/spot/gallery

KARTA ENDPOINTU API	
Identyfikator:	EP07
Ścieżka:	/public/spot/current-view
Nazwa:	Pobierz listę spotów w aktualnym widoku mapy
Parametry wejściowe:	Brak
Query params:	<ul style="list-style-type: none"> <li>• <b>swLng</b>: double (długość geograficzna lewego dolnego rogu)</li> <li>• <b>swLat</b>: double (szerokość geograficzna lewego dolnego rogu)</li> <li>• <b>neLng</b>: double (długość geograficzna prawego górnego rogu)</li> <li>• <b>neLat</b>: double (szerokość geograficzna prawego górnego rogu)</li> <li>• <b>name</b>: String (fragment nazwy spotu, domyślnie pusty)</li> <li>• <b>sorting</b>: String (tryb sortowania, domyślnie none)</li> <li>• <b>ratingFrom</b>: double (minimalna ocena, domyślnie 0.0)</li> <li>• <b>page</b>: Integer (numer strony, domyślnie 0)</li> </ul>

Kod(y) statusu odpowiedzi:	200 OK
Dane zwracane:	<ul style="list-style-type: none"> <li>• <b>Page&lt;SearchSpotDto&gt;</b>: stronicowana lista spotów, gdzie każdy element zawiera: <ul style="list-style-type: none"> <li>– <b>id</b>: Long</li> <li>– <b>name</b>: String</li> <li>– <b>rating</b>: Double (0–5)</li> <li>– <b>ratingCount</b>: Integer (liczba ocen)</li> <li>– <b>firstPhoto</b>: String (URL pierwszego zdjęcia)</li> <li>– <b>tags</b>: Set&lt;SpotTagDto&gt; (tagi spotu)</li> <li>– <b>centerPoint</b>: BorderPoint (środek obszaru spotu)</li> </ul> </li> </ul>

**Tabela 7.8:** Karta endpointu: /public/spot/current-view

KARTA ENDPOINTU API	
Identyfikator:	EP08
Ścieżka:	/public/spot/get-spot-basic-weather
Nazwa:	Pobierz podstawowe informacje pogodowe dla wskazanej lokalizacji
Parametry wejściowe:	Brak
Query params:	<ul style="list-style-type: none"> <li>• <b>latitude</b>: double (szerokość geograficzna)</li> <li>• <b>longitude</b>: double (długość geograficzna)</li> </ul>

Kod(y) statusu odpowiedzi:	200 OK
Dane zwracane:	<ul style="list-style-type: none"> <li>• <b>temperature</b>: Double (temperatura)</li> <li>• <b>weatherCode</b>: int (kod warunków pogodowych)</li> <li>• <b>windSpeed</b>: Double (prędkość wiatru)</li> <li>• <b>isDay</b>: boolean (czy jest dzień)</li> </ul>

Tabela 7.9: Karta endpointu: /public/spot/get-spot-basic-weather

KARTA ENDPOINTU API	
Identyfikator:	EP09
Ścieżka:	/public/spot/get-spot-detailed-weather
Nazwa:	Pobierz szczegółowe informacje pogodowe dla wskazanej lokalizacji
Parametry wejściowe:	Brak
Query params:	<ul style="list-style-type: none"> <li>• <b>latitude</b>: double</li> <li>• <b>longitude</b>: double</li> </ul>
Kod(y) statusu odpowiedzi:	200 OK

<b>Dane zwracane:</b>	<ul style="list-style-type: none"> <li>• <b>temperature:</b> Double</li> <li>• <b>weatherCode:</b> int</li> <li>• <b>precipitationProbability:</b> Double (prawdopodobieństwo opadów)</li> <li>• <b>dewPoint:</b> Double (punkt rosy)</li> <li>• <b>relativeHumidity:</b> Double (wilgotność względna)</li> <li>• <b>isDay:</b> boolean</li> <li>• <b>uvIndexMax:</b> Double (maksymalny indeks UV)</li> </ul>
---------------------------	--

**Tabela 7.10:** Karta endpointu: /public/spot/get-spot-detailed-weather

KARTA ENDPOINTU API	
<b>Identyfikator:</b>	EP10
<b>Ścieżka:</b>	/public/spot/get-spot-wind-speeds
<b>Nazwa:</b>	Pobierz prędkości wiatru dla spotu na różnych wysokościach
<b>Parametry wejściowe:</b>	Brak
<b>Query params:</b>	<ul style="list-style-type: none"> <li>• <b>latitude:</b> double</li> <li>• <b>longitude:</b> double</li> <li>• <b>spotId:</b> long</li> </ul>
<b>Kod(y) statusu odpowiedzi:</b>	200 OK



<b>Dane zwracane:</b>	<ul style="list-style-type: none"> <li>• <b>windSpeeds100m:</b> Double</li> <li>• <b>windSpeeds200m:</b> Double</li> <li>• <b>windSpeeds300m:</b> Double</li> <li>• <b>windSpeeds500m:</b> Double</li> <li>• <b>windSpeeds750m:</b> Double</li> <li>• <b>windSpeeds1000m:</b> Double</li> </ul>
-----------------------	---

**Tabela 7.11:** Karta endpointu: `/public/spot/get-spot-wind-speeds`

### Wyszukiwarka spotów

KARTA ENDPOINTU API	
<b>Identyfikator:</b>	EP11
<b>Ścieżka:</b>	<code>/public/spot/most-popular</code>
<b>Nazwa:</b>	Pobierz 18 najpopularniejszych spotów
<b>Parametry wejściowe:</b>	Brak
<b>Query params:</b>	Brak
<b>Kod(y) statusu odpowiedzi:</b>	200 OK
<b>Dane zwracane:</b>	<ul style="list-style-type: none"> <li>• <b>items:</b> List&lt;TopRatedSpotDto&gt;, każdy element zawiera: <ul style="list-style-type: none"> <li>– <b>id:</b> Long</li> <li>– <b>name:</b> String</li> <li>– <b>city:</b> String</li> <li>– <b>imageUrl:</b> String</li> </ul> </li> </ul>

**Tabela 7.12:** Karta endpointu: `/public/spot/most-popular`

KARTA ENDPOINTU API	
<b>Identyfikator:</b>	EP12
<b>Ścieżka:</b>	/public/spot/search/home-page
<b>Nazwa:</b>	Wyszukaj spoty na stronie głównej na podstawie lokalizacji
<b>Parametry wejściowe:</b>	Brak
<b>Query params:</b>	<ul style="list-style-type: none"> <li>• <b>country:</b> String (opcjonalnie, kraj)</li> <li>• <b>region:</b> String (opcjonalnie, region)</li> <li>• <b>city:</b> String (opcjonalnie, miasto)</li> <li>• <b>userLongitude:</b> Double (opcjonalnie, długość geograficzna użytkownika)</li> <li>• <b>userLatitude:</b> Double (opcjonalnie, szerokość geograficzna użytkownika)</li> <li>• <b>page:</b> Integer (numer strony, domyślnie 0)</li> <li>• <b>size:</b> Integer (rozmiar strony, domyślnie 20)</li> </ul>
<b>Kod(y) statusu odpowiedzi:</b>	200 OK

<b>Dane zwracane:</b>	<ul style="list-style-type: none"> <li>• <b>items:</b> List&lt;HomePageSpotDto&gt;, każdy element: <ul style="list-style-type: none"> <li>– <b>id:</b> Long</li> <li>– <b>name:</b> String</li> <li>– <b>rating:</b> Double</li> <li>– <b>ratingCount:</b> Integer</li> <li>– <b>firstPhoto:</b> String (URL)</li> <li>– <b>tags:</b> Set&lt;SpotTagDto&gt;</li> <li>– <b>centerPoint:</b> BorderPoint</li> <li>– <b>city:</b> String</li> <li>– <b>distanceToUser:</b> Double (odległość od użytkownika, jeśli dostępna)</li> </ul> </li> <li>• <b>hasNext:</b> boolean (czy istnieje kolejna strona wyników)</li> </ul>
---------------------------	---

**Tabela 7.13:** Karta endpointu: /public/spot/search/home-page

KARTA ENDPOINTU API	
<b>Identyfikator:</b>	EP13
<b>Ścieżka:</b>	/public/spot/search/home-page/locations
<b>Nazwa:</b>	Pobierz listę podpowiedzi lokalizacji dla wyszukiwarki na stronie głównej
<b>Parametry wejściowe:</b>	Brak
<b>Query params:</b>	<ul style="list-style-type: none"> <li>• <b>query:</b> String (frazę wyszukiwania)</li> <li>• <b>type:</b> String (typ lokalizacji, kraj/region/miasto)</li> </ul>

Kod(y) statusu odpowiedzi:	200 OK
Dane zwracane:	<ul style="list-style-type: none"> <li>• <b>List&lt;String&gt;</b>: lista podpowiedzi (nazwy lokalizacji)</li> </ul>

Tabela 7.14: Karta endpointu: /public/spot/search/home-page/locations

KARTA ENDPOINTU API	
Identyfikator:	EP14
Ścieżka:	/public/spot/search/home-page/advance
Nazwa:	Wyszukaj spoty na stronie głównej (zaawansowane filtrowanie)
Parametry wejściowe:	Brak
Query params:	<ul style="list-style-type: none"> <li>• <b>city</b>: String (opcjonalnie, miasto)</li> <li>• <b>tags</b>: List&lt;String&gt; (opcjonalnie, lista tagów)</li> <li>• <b>userLongitude</b>: Double (opcjonalnie, długość geograficzna użytkownika)</li> <li>• <b>userLatitude</b>: Double (opcjonalnie, szerokość geograficzna użytkownika)</li> <li>• <b>sort</b>: SpotSortType (opcjonalnie, typ sortowania)</li> <li>• <b>filter</b>: SpotRatingFilterType (opcjonalnie, filtr po ocenie)</li> <li>• <b>page</b>: Integer (numer strony, domyślnie 0)</li> <li>• <b>size</b>: Integer (rozmiar strony, domyślnie 20)</li> </ul>
Kod(y) statusu odpowiedzi:	200 OK

<b>Dane zwracane:</b>	<ul style="list-style-type: none"> <li>• <b>items:</b> List&lt;HomePageSpotDto&gt;, każdy element: <ul style="list-style-type: none"> <li>– <b>id:</b> Long</li> <li>– <b>name:</b> String</li> <li>– <b>rating:</b> Double</li> <li>– <b>ratingCount:</b> Integer</li> <li>– <b>firstPhoto:</b> String (URL)</li> <li>– <b>tags:</b> Set&lt;SpotTagDto&gt;</li> <li>– <b>centerPoint:</b> BorderPoint</li> <li>– <b>city:</b> String</li> <li>– <b>distanceToUser:</b> Double (odległość od użytkownika, jeśli dostępna)</li> </ul> </li> <li>• <b>hasNext:</b> boolean (czy istnieje kolejna strona wyników)</li> </ul>
---------------------------	---

**Tabela 7.15:** Karta endpointu: /public/spot/search/home-page/advance

#### Komentarze do spotów

KARTA ENDPOINTU API	
<b>Identyfikator:</b>	EP15
<b>Ścieżka:</b>	/public/spot/{spotId}/comments
<b>Nazwa:</b>	Pobierz stronicowaną listę komentarzy dla wskazanego spotu
<b>Parametry wejściowe:</b>	<ul style="list-style-type: none"> <li>• <b>spotId:</b> Long (ID spotu w ścieżce URL)</li> </ul>
<b>Query params:</b>	<ul style="list-style-type: none"> <li>• <b>page:</b> Integer (numer strony, domyślnie 0; rozmiar strony = 2)</li> </ul>

<b>Kod(y) statusu odpowiedzi:</b>	200 OK, 404 Not Found, 401 Unauthorized
<b>Dane zwracane:</b>	<ul style="list-style-type: none"> <li>• <b>Page&lt;SpotCommentDto&gt;</b>: stronicowana lista komentarzy, każdy element zawiera: <ul style="list-style-type: none"> <li>– <b>id</b>: Long (ID komentarza)</li> <li>– <b>author</b>: SpotCommentAuthorDto (dane autora)</li> <li>– <b>text</b>: String (treść komentarza)</li> <li>– <b>rating</b>: Double (ocena komentarza/spotu, 0–5)</li> <li>– <b>upvotes</b>: Integer (liczba głosów pozytywnych)</li> <li>– <b>downvotes</b>: Integer (liczba głosów negatywnych)</li> <li>– <b>publishDate</b>: LocalDateTime (data publikacji)</li> <li>– <b>isUpVoted</b>: Boolean (czy bieżący użytkownik zagłosował w górę)</li> <li>– <b>isDownVoted</b>: Boolean (czy bieżący użytkownik zagłosował w dół)</li> <li>– <b>numberOfMedia</b>: Integer (liczba dołączonych plików)</li> <li>– <b>mediaList</b>: List&lt;SpotCommentMediaDto&gt; (lista pierwszych plików komentarza)</li> </ul> </li> </ul>

**Tabela 7.16:** Karta endpointu: `/public/spot/{spotId}/comments`

KARTA ENDPOINTU API	
<b>Identyfikator:</b>	EP16
<b>Ścieżka:</b>	<code>/public/spot/{spotId}/comments/{commentId}</code>

<b>Nazwa:</b>	Pobierz pełną listę mediów powiązanych z komentarzem
<b>Parametry wejściowe:</b>	<ul style="list-style-type: none"> <li>• <b>spotId</b>: Long (ID spotu w ścieżce URL)</li> <li>• <b>commentId</b>: Long (ID komentarza w ścieżce URL)</li> </ul>
<b>Query params:</b>	Brak
<b>Kod(y) statusu odpowiedzi:</b>	200 OK, 404 Not Found
<b>Dane zwracane:</b>	<ul style="list-style-type: none"> <li>• <b>List&lt;SpotCommentMediaDto&gt;</b>: lista mediów komentarza, każdy element: <ul style="list-style-type: none"> <li>– <b>id</b>: Long (ID pliku)</li> <li>– <b>url</b>: String (URL pliku)</li> <li>– <b>genericMediaType</b>: GenericMediaType (typ pliku, PHOTO, VIDEO)</li> </ul> </li> </ul>

**Tabela 7.17:** Karta endpointu: `/public/spot/{spotId}/comments/{commentId}`

KARTA ENDPOINTU API	
<b>Identyfikator:</b>	EP17
<b>Ścieżka:</b>	<code>/spot/{spotId}/comments</code>
<b>Nazwa:</b>	Dodaj nowy komentarz do wskazanego spotu
<b>Parametry wejściowe:</b>	<ul style="list-style-type: none"> <li>• <b>spotId</b>: Long (ID spotu w ścieżce URL)</li> <li>• <b>body</b>: SpotCommentAddDto, zawiera: <ul style="list-style-type: none"> <li>– <b>text</b>: String</li> <li>– <b>rating</b>: Double</li> <li>– <b>mediaFiles</b>: List&lt;MultipartFile&gt;</li> </ul> </li> </ul>

<b>Query params:</b>	Brak
<b>Kod(y) statusu odpowiedzi:</b>	201 Created, 404 Not Found, 401 Unauthorized, 422 Unprocessable Entity
<b>Dane zwracane:</b>	Brak (pusta odpowiedź)

**Tabela 7.18:** Karta endpointu: `/spot/{spotId}/comments`

KARTA ENDPOINTU API	
<b>Identyfikator:</b>	EP18
<b>Ścieżka:</b>	<code>/spot/comments/{commentId}/vote</code>
<b>Nazwa:</b>	Oddaj głos na komentarz (góra/dół)
<b>Parametry wejściowe:</b>	<ul style="list-style-type: none"> <li>• <b>commentId:</b> Long (ID komentarza w ścieżce URL)</li> </ul>
<b>Query params:</b>	<ul style="list-style-type: none"> <li>• <b>isUpvote:</b> boolean (true = głos w górę, false = głos w dół)</li> </ul>
<b>Kod(y) statusu odpowiedzi:</b>	200 OK, 401 Unauthorized, 404 Not Found, 409 Conflict, 403 Forbidden
<b>Dane zwracane:</b>	Brak (pusta odpowiedź)

**Tabela 7.19:** Karta endpointu: `/spot/comments/{commentId}/vote`



KARTA ENDPOINTU API	
<b>Identyfikator:</b>	EP19
<b>Ścieżka:</b>	/spot/comments/vote-type
<b>Nazwa:</b>	Pobierz informację, jak bieżący użytkownik zagłosował na komentarz
<b>Parametry wejściowe:</b>	Brak
<b>Query params:</b>	<ul style="list-style-type: none"> <li>• <b>commentId:</b> Long (ID komentarza)</li> </ul>
<b>Kod(y) statusu odpowiedzi:</b>	200 OK, 404 Not Found, 401 Unauthorized
<b>Dane zwracane:</b>	<ul style="list-style-type: none"> <li>• <b>voteInfo:</b> SpotCommentVoteType (typ oddanego głosu, UPVOTE, DOWNVOTE, NONE)</li> </ul>

Tabela 7.20: Karta endpointu: /spot/comments/vote-type

## Forum – posty

KARTA ENDPOINTU API	
<b>Identyfikator:</b>	EP20
<b>Ścieżka:</b>	/public/post/{postId}
<b>Nazwa:</b>	Pobierz szczegółowe informacje o poście
<b>Parametry wejściowe:</b>	<ul style="list-style-type: none"> <li>• <b>postId:</b> Long (ID posta w ścieżce URL)</li> </ul>
<b>Query params:</b>	Brak

<b>Kod(y) statusu odpowiedzi:</b>	200 OK, 404 Not Found
<b>Dane zwracane:</b>	<ul style="list-style-type: none"> <li>• <b>PostDetailsDto</b>, zawiera: <ul style="list-style-type: none"> <li>– <b>id</b>: Long</li> <li>– <b>title</b>: String (tytuł posta)</li> <li>– <b>content</b>: String (pełna treść posta)</li> <li>– <b>category</b>: ForumCategoryDto</li> <li>– <b>tags</b>: List&lt;ForumTagDto&gt;</li> <li>– <b>author</b>: AuthorDto (dane autora)</li> <li>– <b>isAuthor</b>: Boolean (czy bieżący użytkownik jest autorem)</li> <li>– <b>isFollowed</b>: Boolean (czy bieżący użytkownik obserwuje post)</li> <li>– <b>publishDate</b>: LocalDateTime (data publikacji)</li> <li>– <b>views</b>: Integer (liczba wyświetleń)</li> <li>– <b>upVotes</b>: Integer</li> <li>– <b>downVotes</b>: Integer</li> <li>– <b>isUpVoted</b>: Boolean (czy użytkownik oddał głos w górę)</li> <li>– <b>isDownVoted</b>: Boolean (czy użytkownik oddał głos w dół)</li> <li>– <b>commentsCount</b>: Integer (liczba komentarzy)</li> </ul> </li> </ul>

**Tabela 7.21:** Karta endpointu: /public/post/{postId}

KARTA ENDPOINTU API	
<b>Identyfikator:</b>	EP21
<b>Ścieżka:</b>	/post
<b>Nazwa:</b>	Dodaj nowy post na forum
<b>Parametry wejściowe:</b>	<ul style="list-style-type: none"> <li>• <b>body:</b> PostDto, zawiera: <ul style="list-style-type: none"> <li>– <b>title:</b> String (tytuł posta)</li> <li>– <b>content:</b> String (pełna treść posta)</li> <li>– <b>category:</b> String</li> <li>– <b>tags:</b> List&lt;String&gt;</li> </ul> </li> </ul>
<b>Query params:</b>	Brak
<b>Kod(y) statusu odpowiedzi:</b>	201 Created, 400 Bad Request, 404 Not Found, 409 Conflict, 422 Unprocessable Entity
<b>Dane zwracane:</b>	Brak (pusta odpowiedź)

**Tabela 7.22:** Karta endpointu: /post

KARTA ENDPOINTU API	
<b>Identyfikator:</b>	EP22
<b>Ścieżka:</b>	/post/{postId}
<b>Nazwa:</b>	Usuń wybrany post
<b>Parametry wejściowe:</b>	<ul style="list-style-type: none"> <li>• <b>postId:</b> Long (ID posta w ścieżce URL)</li> </ul>
<b>Query params:</b>	Brak

<b>Kod(y) statusu odpowiedzi:</b>	204 No Content, 401 Unauthorized, 404 Not Found
<b>Dane zwracane:</b>	Brak (pusta odpowiedź)

**Tabela 7.23:** Karta endpointu: /post/{postId}

KARTA ENDPOINTU API	
<b>Identyfikator:</b>	EP23
<b>Ścieżka:</b>	/post/{postId}/vote
<b>Nazwa:</b>	Oddaj głos na post (góra/dół)
<b>Parametry wejściowe:</b>	<ul style="list-style-type: none"> <li>• <b>postId</b>: Long (ID posta w ścieżce URL)</li> </ul>
<b>Query params:</b>	<ul style="list-style-type: none"> <li>• <b>isUpvote</b>: boolean (true = głos w górę, false = głos w dół)</li> </ul>
<b>Kod(y) statusu odpowiedzi:</b>	200 OK, 401 Unauthorized, 404 Not Found
<b>Dane zwracane:</b>	Brak (pusta odpowiedź)

**Tabela 7.24:** Karta endpointu: /post/{postId}/vote

KARTA ENDPOINTU API	
<b>Identyfikator:</b>	EP24

<b>Ścieżka:</b>	/public/categories-tags
<b>Nazwa:</b>	Pobierz listę wszystkich kategorii i tagów forum
<b>Parametry wejściowe:</b>	Brak
<b>Query params:</b>	Brak
<b>Kod(y) statusu odpowiedzi:</b>	200 OK
<b>Dane zwracane:</b>	<ul style="list-style-type: none"> <li>• <b>ForumCategoriesAndTagsDto:</b> <ul style="list-style-type: none"> <li>– <b>categories:</b> List&lt;ForumCategoryDto&gt;, gdzie każdy element zawiera: <ul style="list-style-type: none"> <li>* <b>id:</b> Long</li> <li>* <b>name:</b> String</li> <li>* <b>description:</b> String</li> <li>* <b>colour:</b> String (kolor kategorii)</li> </ul> </li> <li>– <b>tags:</b> List&lt;ForumTagDto&gt;, gdzie każdy element zawiera: <ul style="list-style-type: none"> <li>* <b>id:</b> Long</li> <li>* <b>name:</b> String</li> </ul> </li> </ul> </li> </ul>

**Tabela 7.25:** Karta endpointu: /public/categories-tags

## Forum – komentarze do postów

KARTA ENDPOINTU API	
<b>Identyfikator:</b>	EP25

<b>Ścieżka:</b>	/public/post/{postId}/comments
<b>Nazwa:</b>	Pobierz stronicowaną listę komentarzy posta
<b>Parametry wejściowe:</b>	<ul style="list-style-type: none"> <li>• <b>postId:</b> Long (ID posta w ścieżce URL)</li> </ul>
<b>Query params:</b>	<ul style="list-style-type: none"> <li>• <b>page:</b> Integer (numer strony, domyślnie 0)</li> <li>• <b>size:</b> Integer (liczba komentarzy na stronie, domyślnie 10)</li> <li>• <b>sortBy:</b> PostCommentSortField (pole sortowania, domyślnie PUBLISH_DATE)</li> <li>• <b>sortDirection:</b> SortDirection (kierunek sortowania, domyślnie DESC)</li> </ul>
<b>Kod(y) statusu odpowiedzi:</b>	200 OK, 404 Not Found

Dane zwracane:	<ul style="list-style-type: none"> <li>• <b>Page&lt;PostCommentGeneralDto&gt;</b>, każdy element zawiera: <ul style="list-style-type: none"> <li>– <b>id</b>: Long</li> <li>– <b>content</b>: String (treść komentarza)</li> <li>– <b>upVotes</b>: Integer (liczba głosów w górę)</li> <li>– <b>downVotes</b>: Integer (liczba głosów w dół)</li> <li>– <b>repliesCount</b>: Integer (liczba odpowiedzi)</li> <li>– <b>publishDate</b>: LocalDateTime (data publikacji)</li> <li>– <b>author</b>: AuthorDto (dane autora)</li> <li>– <b>isAuthor</b>: Boolean (czy bieżący użytkownik jest autorem)</li> <li>– <b>isUpVoted</b>: Boolean (czy użytkownik zagłosował w górę)</li> <li>– <b>isDownVoted</b>: Boolean (czy użytkownik zagłosował w dół)</li> <li>– <b>isReply</b>: Boolean (czy komentarz jest odpowiedzią)</li> <li>– <b>isDeleted</b>: Boolean (czy komentarz został usunięty logicznie)</li> </ul> </li> </ul>
----------------	--

**Tabela 7.26:** Karta endpointu: /public/post/{postId}/comments

KARTA ENDPOINTU API	
<b>Identyfikator:</b>	EP26
<b>Ścieżka:</b>	/post/{postId}/comments
<b>Nazwa:</b>	Dodaj nowy komentarz do posta

<b>Parametry wejściowe:</b>	<ul style="list-style-type: none"> <li>• <b>postId</b>: Long (ID posta w ścieżce URL)</li> <li>• <b>body</b>: PostCommentDto, zawiera: <ul style="list-style-type: none"> <li>– <b>content</b>: String</li> </ul> </li> </ul>
<b>Query params:</b>	Brak
<b>Kod(y) statusu odpowiedzi:</b>	201 Created, 400 Bad Request, 404 Not Found, 422 Unprocessable Entity
<b>Dane zwracane:</b>	Brak (pusta odpowiedź)

**Tabela 7.27:** Karta endpointu: `/post/{postId}/comments`

KARTA ENDPOINTU API	
<b>Identyfikator:</b>	EP27
<b>Ścieżka:</b>	<code>/post/comments/{commentId}</code>
<b>Nazwa:</b>	Edytuj istniejący komentarz do posta
<b>Parametry wejściowe:</b>	<ul style="list-style-type: none"> <li>• <b>commentId</b>: Long (ID komentarza w ścieżce URL)</li> <li>• <b>body</b>: PostCommentDto, zawiera: <ul style="list-style-type: none"> <li>– <b>content</b>: String</li> </ul> </li> </ul>
<b>Query params:</b>	Brak
<b>Kod(y) statusu odpowiedzi:</b>	200 OK, 400 Bad Request, 401 Unauthorized, 403 Forbidden, 404 Not Found, 422 Unprocessable Entity
<b>Dane zwracane:</b>	Brak (pusta odpowiedź)



Tabela 7.28: Karta endpointu: /post/comments/{commentId}

KARTA ENDPOINTU API	
<b>Identyfikator:</b>	EP28
<b>Ścieżka:</b>	/post/comments/{commentId}/vote
<b>Nazwa:</b>	Oddaj głos na komentarz (góra/dół)
<b>Parametry wejściowe:</b>	<ul style="list-style-type: none"> <li>• <b>commentId:</b> Long (ID komentarza w ścieżce URL)</li> </ul>
<b>Query params:</b>	<ul style="list-style-type: none"> <li>• <b>isUpvote:</b> boolean (true = głos w górę, false = głos w dół)</li> </ul>
<b>Kod(y) statusu odpowiedzi:</b>	200 OK, 403 Forbidden, 404 Not Found
<b>Dane zwracane:</b>	Brak (pusta odpowiedź)

Tabela 7.29: Karta endpointu: /post/comments/{commentId}/vote

KARTA ENDPOINTU API	
<b>Identyfikator:</b>	EP29
<b>Ścieżka:</b>	/comments/{commentId}/replies
<b>Nazwa:</b>	Dodaj odpowiedź na komentarz

<b>Parametry wejściowe:</b>	<ul style="list-style-type: none"> <li>• <b>commentId</b>: Long (ID komentarza nadrzędnego w ścieżce URL)</li> <li>• <b>body</b>: PostCommentDto, zawiera: <ul style="list-style-type: none"> <li>– <b>content</b>: String</li> </ul> </li> </ul>
<b>Query params:</b>	Brak
<b>Kod(y) statusu odpowiedzi:</b>	201 Created, 400 Bad Request, 404 Not Found, 409 Conflict, 422 Unprocessable Entity
<b>Dane zwracane:</b>	Brak (pusta odpowiedź)

**Tabela 7.30:** Karta endpointu: `/comments/{commentId}/replies`

### Konto użytkownika – rejestracja, logowanie, hasło

KARTA ENDPOINTU API	
<b>Identyfikator:</b>	EP30
<b>Ścieżka:</b>	<code>/public/account/register</code>
<b>Nazwa:</b>	Zarejestruj nowego użytkownika
<b>Parametry wejściowe:</b>	<ul style="list-style-type: none"> <li>• <b>body</b>: UserRegisterDto, zawiera: <ul style="list-style-type: none"> <li>– <b>username</b>: String</li> <li>– <b>email</b>: String</li> <li>– <b>password</b>: String</li> </ul> </li> </ul>
<b>Query params:</b>	Brak

<b>Kod(y) statusu odpowiedzi:</b>	201 Created, 401 Unauthorized, 409 Conflict, 422 Unprocessable Entity, 500 Internal Server Error
<b>Dane zwracane:</b>	<ul style="list-style-type: none"> <li>• <b>body:</b> String (komunikat)</li> <li>• JWT tokeny ustawione w ciasteczkach HTTP-only</li> </ul>

**Tabela 7.31:** Karta endpointu: /public/account/register

KARTA ENDPOINTU API	
<b>Identyfikator:</b>	EP31
<b>Ścieżka:</b>	/public/account/login
<b>Nazwa:</b>	Zaloguj użytkownika
<b>Parametry wejściowe:</b>	<ul style="list-style-type: none"> <li>• <b>body:</b> UserLoginDto, zawiera: <ul style="list-style-type: none"> <li>– <b>username:</b> String</li> <li>– <b>password:</b> String</li> </ul> </li> </ul>
<b>Query params:</b>	Brak
<b>Kod(y) statusu odpowiedzi:</b>	200 OK, 401 Unauthorized, 422 Unprocessable Entity
<b>Dane zwracane:</b>	<ul style="list-style-type: none"> <li>• Brak (pusta odpowiedź w body)</li> <li>• JWT tokeny zwrócone w ciasteczkach HTTP-only</li> </ul>

**Tabela 7.32:** Karta endpointu: /public/account/login

KARTA ENDPOINTU API
---------------------

<b>Identyfikator:</b>	EP32
<b>Ścieżka:</b>	/public/account/forgot-password
<b>Nazwa:</b>	Rozpocznij procedurę resetu hasła (wyślij link na e-mail)
<b>Parametry wejściowe:</b>	<ul style="list-style-type: none"> <li>• <b>body:</b> String (adres e-mail użytkownika w treści żądania)</li> </ul>
<b>Query params:</b>	Brak
<b>Kod(y) statusu odpowiedzi:</b>	200 OK, 404 Not Found, 422 Unprocessable Entity, 500 Internal Server Error
<b>Dane zwracane:</b>	<ul style="list-style-type: none"> <li>• <b>body:</b> String (komunikat)</li> <li>• Link resetujący hasło wysłany na podany adres e-mail</li> </ul>

**Tabela 7.33:** Karta endpointu: /public/account/forgot-password

KARTA ENDPOINTU API	
<b>Identyfikator:</b>	EP33
<b>Ścieżka:</b>	/public/account/set-new-password
<b>Nazwa:</b>	Ustaw nowe hasło użytkownika na podstawie tokenu resetującego
<b>Parametry wejściowe:</b>	<ul style="list-style-type: none"> <li>• <b>body:</b> UserPasswordResetDto, zawiera: <ul style="list-style-type: none"> <li>– <b>token:</b> String (UUID – token resetu hasła)</li> <li>– <b>password:</b> String</li> </ul> </li> </ul>
<b>Query params:</b>	Brak

<b>Kod(y) statusu odpowiedzi:</b>	200 OK, 400 Bad Request, 404 Not Found, 422 Unprocessable Entity
<b>Dane zwracane:</b>	<ul style="list-style-type: none"> <li>• <b>body:</b> String (komunikat)</li> </ul>

**Tabela 7.34:** Karta endpointu: /public/account/set-new-password

KARTA ENDPOINTU API	
<b>Identyfikator:</b>	EP34
<b>Ścieżka:</b>	/account/check
<b>Nazwa:</b>	Sprawdź, czy użytkownik jest uwierzytelniony
<b>Parametry wejściowe:</b>	Brak
<b>Query params:</b>	Brak
<b>Kod(y) statusu odpowiedzi:</b>	200 OK, 401 Unauthorized, 403 Forbidden
<b>Dane zwracane:</b>	<ul style="list-style-type: none"> <li>• Brak (pusta odpowiedź; sam status informuje o uwierzytelnieniu)</li> </ul>

**Tabela 7.35:** Karta endpointu: /account/check

### Upload multimediiów (Azure Blob)

KARTA ENDPOINTU API	
<b>Identyfikator:</b>	EP74

<b>Ścieżka:</b>	/public/upload/media
<b>Nazwa:</b>	Prześlij plik multimedialny do Azure Blob Storage
<b>Parametry wejściowe:</b>	Brak
<b>Query params:</b>	<ul style="list-style-type: none"> <li>• <b>file:</b> MultipartFile (plik multimedialny)</li> <li>• <b>containerName:</b> String (nazwa kontenera docelowego)</li> </ul>
<b>Kod(y) statusu odpowiedzi:</b>	200 OK, 404 Not Found, 413 Payload Too Large, 415 Unsupported Media Type, 500 Internal Server Error
<b>Dane zwracane:</b>	<ul style="list-style-type: none"> <li>• <b>body:</b> String (URL przesłanego zasobu w Azure Blob Storage)</li> </ul>

**Tabela 7.36:** Karta endpointu: /public/upload/media

## GIF-y (Tenor) – integracja czatu

KARTA ENDPOINTU API	
<b>Identyfikator:</b>	EP35
<b>Ścieżka:</b>	/gifs/trending
<b>Nazwa:</b>	Pobierz listę trendujących kategorii GIF-ów z Tenor
<b>Parametry wejściowe:</b>	Brak
<b>Query params:</b>	Brak
<b>Kod(y) statusu odpowiedzi:</b>	200 OK, 500 Internal Server Error

<b>Dane zwracane:</b>	<ul style="list-style-type: none"> <li>• <b>items:</b> List&lt;TenorGifCategoryDto&gt;, każdy element zawiera: <ul style="list-style-type: none"> <li>– <b>searchTerm:</b> String (frazę wyszukiwania powiązana z kategorią)</li> <li>– <b>path:</b> String (ścieżka kategorii w Tenor)</li> <li>– <b>gifUrl:</b> String</li> </ul> </li> </ul>
-----------------------	---

**Tabela 7.37:** Karta endpointu: /gifs/trending

KARTA ENDPOINTU API	
<b>Identyfikator:</b>	EP36
<b>Ścieżka:</b>	/gifs/search
<b>Nazwa:</b>	Wyszukaj GIF-y po frazie tekstowej
<b>Parametry wejściowe:</b>	Brak
<b>Query params:</b>	<ul style="list-style-type: none"> <li>• <b>searchPhrase:</b> String (frazę wyszukiwania)</li> <li>• <b>next:</b> String (token paginacji zwrócony z poprzedniego wywołania; dla pierwszego zapytania może być pusty)</li> </ul>
<b>Kod(y) statusu odpowiedzi:</b>	200 OK, 500 Internal Server Error
<b>Dane zwracane:</b>	<ul style="list-style-type: none"> <li>• <b>body:</b> TenorGifSearchWrapperDto, zawiera: <ul style="list-style-type: none"> <li>– <b>gifs:</b> List&lt;TenorGifSearchDto&gt;, każdy element: <ul style="list-style-type: none"> <li>* <b>url:</b> String (URL GIF-a)</li> </ul> </li> <li>– <b>next:</b> String (token do pobrania kolejnej strony wyników)</li> </ul> </li> </ul>

**Tabela 7.38:** Karta endpointu: /gifs/search

## Czat – REST API

KARTA ENDPOINTU API	
Identyfikator:	EP37
Ścieżka:	/chats/{chatId}/messages
Nazwa:	Pobierz stronicowane wiadomości dla wybranego czatu
Parametry wejściowe:	<ul style="list-style-type: none"><li>• <b>chatId</b>: Long (identyfikator czatu)</li></ul>
Query params:	<ul style="list-style-type: none"><li>• <b>pageParam</b>: Integer (numer strony wiadomości, domyślnie 1 – pierwsza strona po wstępnym pobraniu)</li><li>• <b>numberOfMessagesPerPage</b>: Integer (liczba wiadomości na stronę, domyślnie 20)</li></ul>
Kod(y) statusu odpowiedzi:	200 OK, 401 Unauthorized, 404 Not Found



<b>Dane zwracane:</b>	<ul style="list-style-type: none"> <li>• <b>body:</b> ChatMessageDtoSlice, zawiera: <ul style="list-style-type: none"> <li>– <b>messages:</b> List&lt;ChatMessageDto&gt;, każdy element: <ul style="list-style-type: none"> <li>* <b>id:</b> Long</li> <li>* <b>sender:</b> ChatMessageSenderDto</li> <li>* <b>sentAt:</b> LocalDateTime</li> <li>* <b>content:</b> String (treść wiadomości; dla wiadomości plikowych może być pusty)</li> <li>* <b>chatId:</b> Long</li> <li>* <b>attachedFiles:</b> List&lt;ChatMessageAttachedFileDto&gt;</li> </ul> </li> <li>– <b>hasNextSlice:</b> Boolean (czy istnieje kolejna „strona” wiadomości)</li> <li>– <b>numberOfMessages:</b> Integer (liczba wiadomości w tej odpowiedzi)</li> <li>– <b>sliceNumber:</b> Integer (numer bieżącej „porcji”)</li> </ul> </li> </ul>
---------------------------	---

**Tabela 7.39:** Karta endpointu: /chats/{chatId}/messages

KARTA ENDPOINTU API	
<b>Identyfikator:</b>	EP38
<b>Ścieżka:</b>	/chats/get-or-create-private-chat
<b>Nazwa:</b>	Pobierz istniejący lub utwórz nowy prywatny czat z użytkownikiem
<b>Parametry wejściowe:</b>	Brak

<b>Query params:</b>	<ul style="list-style-type: none"> <li>• <b>receiverUsername:</b> String (nazwa użytkownika, z którym chcemy rozmawiać)</li> <li>• <b>chatId:</b> Long (opcjonalnie, ID istniejącego czatu – jeśli znany)</li> </ul>
<b>Kod(y) statusu odpowiedzi:</b>	200 OK, 401 Unauthorized, 404 Not Found
<b>Dane zwracane:</b>	<ul style="list-style-type: none"> <li>• <b>body:</b> ChatDto, zawiera: <ul style="list-style-type: none"> <li>– <b>id:</b> Long (identyfikator czatu)</li> <li>– <b>name:</b> String (nazwa czatu – nazwa grupy lub rozmówcy)</li> <li>– <b>lastMessage:</b> ChatMessageDto (ostatnia wiadomość w czacie)</li> <li>– <b>imgUrl:</b> String (URL avatara czatu)</li> <li>– <b>messages:</b> List&lt;ChatMessageDto&gt; (lista wiadomości zwróconych razem z czatem)</li> <li>– <b>chatType:</b> ChatType (typ czatu: PRIVATE, GROUP)</li> <li>– <b>participants:</b> List&lt;ChatParticipantDto&gt; (lista uczestników czatu)</li> </ul> </li> </ul>

**Tabela 7.40:** Karta endpointu: /chats/get-or-create-private-chat

KARTA ENDPOINTU API	
<b>Identyfikator:</b>	EP39
<b>Ścieżka:</b>	/chats/{chatId}/send-files
<b>Nazwa:</b>	Wyślij jeden lub wiele plików w ramach czatu

<b>Parametry wejściowe:</b>	<ul style="list-style-type: none"> <li>• <b>chatId:</b> Long (identyfikator czatu w ścieżce)</li> <li>• <b>media:</b> List&lt;MultipartFile&gt; (lista załączanych plików)</li> </ul>
<b>Query params:</b>	Brak
<b>Kod(y) statusu odpowiedzi:</b>	201 Created, 401 Unauthorized, 404 Not Found, 413 Payload Too Large, 415 Unsupported Media Type, 500 Internal Server Error
<b>Dane zwracane:</b>	<ul style="list-style-type: none"> <li>• Brak (pusta odpowiedź; wiadomości z plikami pojawiają się w historii czatu)</li> </ul>

**Tabela 7.41:** Karta endpointu: /chats/{chatId}/send-files

KARTA ENDPOINTU API	
<b>Identyfikator:</b>	EP40
<b>Ścieżka:</b>	/chats/create/group
<b>Nazwa:</b>	Utwórz nowy czat grupowy
<b>Parametry wejściowe:</b>	<ul style="list-style-type: none"> <li>• <b>body:</b> CreateGroupChatDto zawiera: <ul style="list-style-type: none"> <li>– <b>usermes:</b> List&lt;String&gt; (lista nazw uczestników chatu)</li> <li>– <b>ownerUsername:</b> String (nazwa właściciela chatu)</li> </ul> </li> </ul>
<b>Query params:</b>	Brak
<b>Kod(y) statusu odpowiedzi:</b>	201 Created, 400 Bad Request, 401 Unauthorized

<b>Dane zwracane:</b>	<ul style="list-style-type: none"> <li>• <b>body:</b> ChatDto, zawiera: <ul style="list-style-type: none"> <li>– <b>id:</b> Long (identyfikator czatu)</li> <li>– <b>name:</b> String (nazwa czatu – nazwa grupy lub rozmówcy)</li> <li>– <b>lastMessage:</b> ChatMessageDto (ostatnia wiadomość w czacie)</li> <li>– <b>imgUrl:</b> String (URL avatara czatu)</li> <li>– <b>messages:</b> List&lt;ChatMessageDto&gt; (lista wiadomości zwróconych razem z czatem)</li> <li>– <b>chatType:</b> ChatType (typ czatu, PRIVATE, GROUP)</li> <li>– <b>participants:</b> List&lt;ChatParticipantDto&gt; (lista uczestników czatu)</li> </ul> </li> </ul>
-----------------------	--

**Tabela 7.42:** Karta endpointu: /chats/create/group

KARTA ENDPOINTU API	
<b>Identyfikator:</b>	EP41
<b>Ścieżka:</b>	/chats/{chatId}
<b>Nazwa:</b>	Zaktualizuj dane czatu grupowego (nazwa, zdjęcie)
<b>Parametry wejściowe:</b>	<ul style="list-style-type: none"> <li>• <b>chatId:</b> Long (identyfikator czatu grupowego)</li> <li>• <b>updateGroupChatDto:</b> UpdateGroupChatDto (wysyłany jako multipart/form-data, zawiera dane do zmiany, nowa nazwa, nowe zdjęcie)</li> </ul>
<b>Query params:</b>	Brak

<b>Kod(y) statusu odpowiedzi:</b>	200 OK, 401 Unauthorized, 404 Not Found, 413 Payload Too Large, 415 Unsupported Media Type, 500 Internal Server Error
<b>Dane zwracane:</b>	<ul style="list-style-type: none"> <li>• <b>body:</b> UpdatedGroupChatDto, zawiera: <ul style="list-style-type: none"> <li>– <b>newName:</b> String (aktualna nazwa czatu)</li> <li>– <b>newImgUrl:</b> String (aktualny URL obrazka grupy)</li> </ul> </li> </ul>

**Tabela 7.43:** Karta endpointu: /chats/{chatId}

KARTA ENDPOINTU API	
<b>Identyfikator:</b>	EP42
<b>Ścieżka:</b>	/chats/group-chat/add/search/{chatId}
<b>Nazwa:</b>	Wyszukaj potencjalnych użytkowników do dodania do czatu grupowego
<b>Parametry wejściowe:</b>	<ul style="list-style-type: none"> <li>• <b>chatId:</b> Long (identyfikator czatu grupowego)</li> </ul>
<b>Query params:</b>	<ul style="list-style-type: none"> <li>• <b>query:</b> String (frazę wyszukiwania po nazwie użytkownika)</li> <li>• <b>page:</b> Integer (numer strony, domyślnie 0)</li> <li>• <b>size:</b> Integer (liczba wyników na stronę, domyślnie 20)</li> </ul>
<b>Kod(y) statusu odpowiedzi:</b>	200 OK, 401 Unauthorized, 404 Not Found

<b>Dane zwracane:</b>	<ul style="list-style-type: none"> <li>• <b>body:</b> SimpleSliceDto&lt;PotentialChatMemberDto&gt;, zawiera: <ul style="list-style-type: none"> <li>– <b>hasNext:</b> boolean (czy istnieje kolejna „strona” wyników)</li> <li>– <b>collection:</b> Collection&lt;PotentialChatMemberDto&gt;, każdy element: <ul style="list-style-type: none"> <li>* <b>username:</b> String (nazwa użytkownika)</li> <li>* <b>profileImg:</b> String (URL zdjęcia profilowego)</li> </ul> </li> </ul> </li> </ul>
---------------------------	---

**Tabela 7.44:** Karta endpointu: /chats/group-chat/add/search/{chatId}

### 7.1.3 Integracja z bazą danych

W aplikacji wykorzystano relacyjną bazę danych PostgreSQL, która w środowisku deweloperskim uruchamiana jest jako kontener w aplikacji Docker. Komunikacja backendu z bazą danych odbywa się z wykorzystaniem wzorca Repository oraz bibliotek oferowanych przez Spring Boot, co umożliwia efektywne zarządzanie danymi oraz utrzymanie spójności warstwy dostępu do danych.

W systemie zaimplementowano zestaw najistotniejszych tabel, które opisano poniżej:

- **chat-invitations** — przechowuje zaproszenia do czatów wysyłane użytkownikom.
- **chat-message-attached-file** — przechowuje pliki dołączone do wiadomości w czatach.
- **chat-messages** — zapisuje wiadomości wysyłane w czatach.
- **chat-participants** — zawiera informacje o uczestnikach poszczególnych czatów.
- **chats** — lista czatów dostępnych w systemie.

- **favorite-spots** — informacje o miejscach (spotach) oznaczonych jako ulubione przez użytkowników.
- **forum-categories** — kategorie, do których przypisywane są posty na forum.
- **forum-tags** — tagi przypisywane postom na forum.
- **friendships** — relacje znajomości między użytkownikami.
- **media** — ogólne media przesyłane przez użytkowników na forum (zdjęcia, filmy).
- **post-comment-down-votes** — przechowuje „minusy” nadawane komentarzom do postów.
- **post-comment-reports** — raporty zgłaszane przez użytkowników wobec komentarzy.
- **post-comment-up-votes** — przechowuje „plusy” nadawane komentarzom do postów.
- **post-comments** — komentarze użytkowników do postów.
- **post-down-votes** — „minusy” nadawane postom.
- **post-followers** — informacje o użytkownikach obserwujących dany post.
- **post-reports** — raporty zgłaszane wobec postów.
- **post-tags** — tagi przypisane do konkretnych postów.
- **post-up-votes** — „plusy” nadawane postom.
- **posts** — posty tworzone przez użytkowników.
- **spot-comment-down-votes** — „minusy” nadawane komentarzom do spotów.
- **spot-comment-media** — pliki multimedialne dołączone do komentarzy przy spotach.

- **spot-comment-up-votes** — „plusy” nadawane komentarzom do spotów.
- **spot-comments** — komentarze użytkowników do spotów.
- **spot-media** — pliki multimedialne związane z konkretnymi spotami.
- **spots** — baza spotów w systemie.
- **spots-tags** — tagi przypisane do poszczególnych spotów.
- **tags-od-spots** — alternatywna tabela z tagami dla spotów.
- **user-followed-posts** — lista postów śledzonych przez użytkowników.
- **user-followers** — relacje obserwujących użytkowników.
- **user-liked-spot-media** — informacja o polubieniach mediów powiązanych ze spotami.
- **users** — dane użytkowników systemu.

#### 7.1.4 Obsługa uwierzytelnienia

#### 7.1.5 Konteneryzacja

### 7.2 Implementacja frontendu

W niniejszym rozdziale przedstawiono proces implementacji części [frontendowej](#) aplikacji.

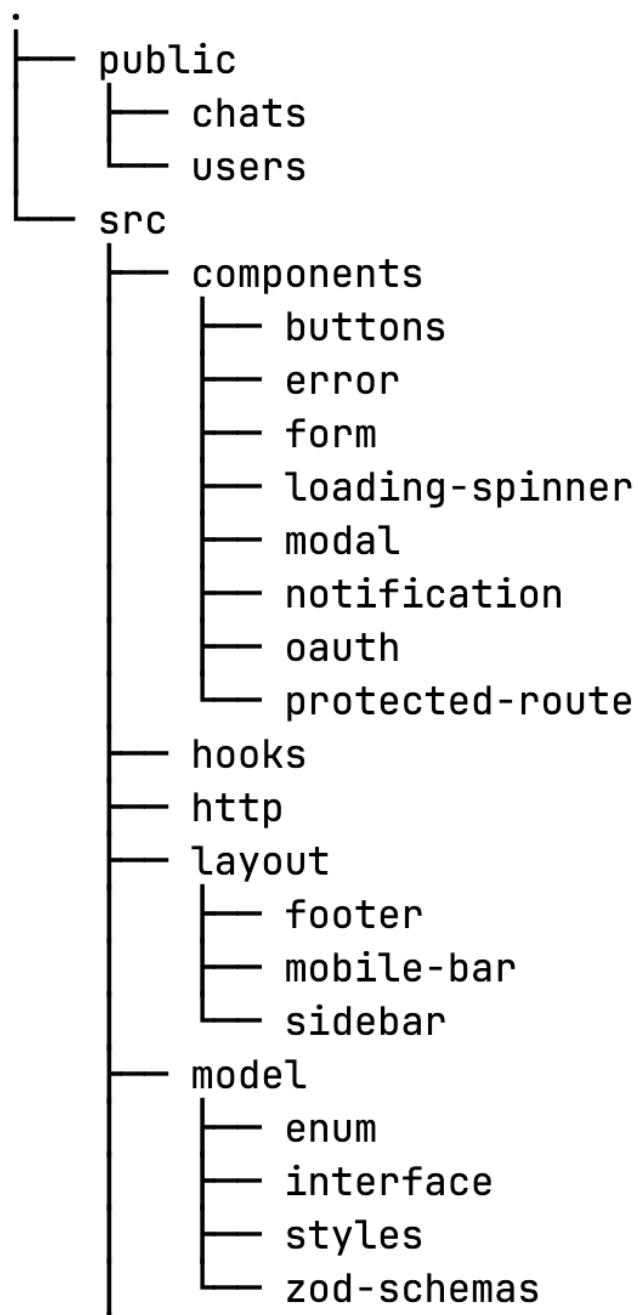
#### 7.2.1 Struktura aplikacji

W niniejszym podrozdziale przedstawiona została struktura aplikacji [frontendowej](#) oraz organizację jej kluczowych elementów.

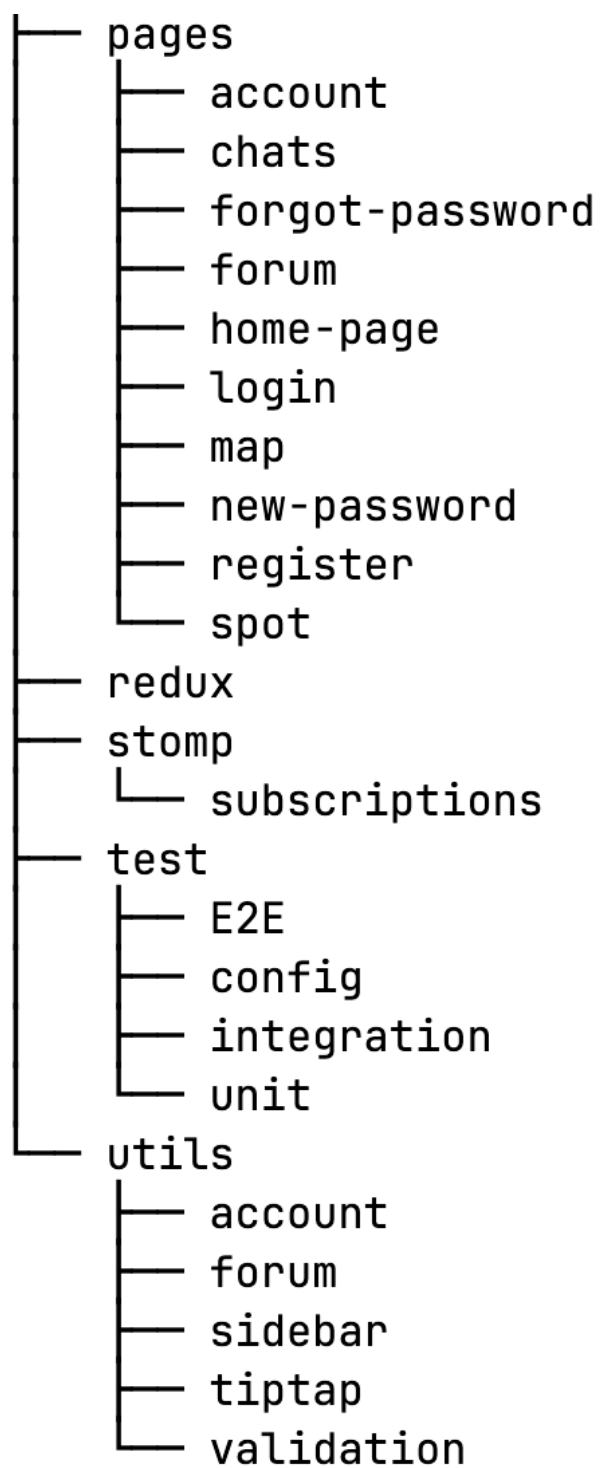
Architekturę aplikacji [frontendowej](#) zaprojektowano w strukturze [Folder by type](#), która polega na podziale kodu według typu zasobu (komponenty, strony,



modele itd.). Każdy plik znajduje się w katalogu odpowiadającym jego przeznaczeniu, co przedstawiono na rysunkach 7.3 oraz 7.4.



Rysunek 7.3: Struktura katalogów (1)



Rysunek 7.4: Struktura katalogów (2)

Głównym elementem aplikacji jest mechanizm routingu oparty na [Bibliotece React Router](#). Definiuje on ścieżki do poszczególnych funkcjonalności aplikacji. Dzięki temu możliwa jest płynna nawigacja między różnymi widokami bez konieczności przeładowywania strony.

```
const router : Router = createBrowserRouter([
  {
    path: "/",
    element: <Layout />,
    errorElement: <Error error={undefined} />,
    children: [
      {
        index: true,
        element: <HomePage />,
      },
      {
        path: "advanced",
        element: <AdvanceHomePage />,
      },
      {
        path: "account",
        children: [ 11 elements... ],
      },
      {
        path: "register",
        element: <Register />,
      },
      {
        path: "login",
        element: <Login />,
      },
      {
        path: "forgot-password",
        element: <ForgotPassword />,
      },
    ],
  },
])
```

Rysunek 7.5: Implementacja routera (1)

```

    {
      path: "new-password",
      element: <NewPassword />,
    },
    {
      path: "forum",
      element: <Forum />,
    },
    {
      path: "forum/:postId/:slugTitle?",
      element: <ForumThread />,
    },
    {
      path: "map",
      element: <MapPage />,
    },
    {
      path: "chat",
      element: (
        <ProtectedRoute>
          <ChatsPage />
        </ProtectedRoute>
      ),
    },
  ],
},
]);

export default router; Show usages  Adam Langmesser

```

Rysunek 7.6: Implementacja routera (2)

W projekcie zastosowano również wzorzec [protected route](#), który służy do zabezpieczania wybranych tras przed dostępem użytkowników niezalogowanych. W pliku `router.tsx`, znajdującym się w głównym katalogu projektu, w konfiguracji przekazywanej do funkcji `createBrowserRouter` (rysunki 7.5 oraz 7.6), wybrane ścieżki opakowano w komponent `ProtectedRoute`. Komponent ten pełni

rolę bramki (rysunek 7.7).

Przykładem takiej chronionej ścieżki jest trasa `/chat`, prowadząca do modułu czatu dostępnego wyłącznie dla zalogowanych użytkowników. Jeśli niezalogowany użytkownik spróbuje uzyskać dostęp do tej ścieżki, zostanie automatycznie przekierowany na stronę główną.

```
export default function ProtectedRoute({ children }) { Show usages  Mredosz
  const isLoggedIn = useSelector((state) => state.account.isLoggedIn);

  return isLoggedIn ? children : <Navigate to="/" />;
}
```

Rysunek 7.7: Implementacja komponentu bramki (`ProtectedRoute`)

### 7.2.2 Zarządzanie stanem i przepływ danych

W niniejszym podrozdziale opisano zastosowane w projekcie podejście do zarządzania [stanem](#) oraz organizację przepływu danych w aplikacji frontendowej.

W projekcie postawiono na zrównoważone podejście do zarządzania [stanem](#). Korzysta się zarówno z lokalnego [stanu](#) komponentów (za pomocą [hooka](#) `useState`) [14], jak i ze [stanu](#) globalnego, utrzymywanego przez [bibliotekę React Redux](#) [15]. Globalny [stan](#) wprowadzono w celu możliwie jak największego ograniczenia przekazywania [propsów](#) w głąb drzewa komponentów oraz uniknąć niepotrzebnych ponownych renderów.

Do przechowywania [stanu](#) lokalnego, ograniczonego tylko do danego komponentu (lub jego najbliższych elementów podrzędnych), wykorzystuje się [hook](#) `useState`. Natomiast efekty uboczne i synchronizację realizuje się za pomocą `useEffect`. W przypadku bardziej złożonej logiki lub potrzeby ponownego wykorzystania kodu powstały [hooki](#) niestandardowe, takie jak `useScreenSize`, `useDarkMode` czy `useClickOutside`. Dzięki temu większość logiki prezentacji wydzielono z warstwy [UI](#), co poprawia czytelność i ułatwia utrzymanie kodu.

Z racji tego, że korzystamy z [reacta](#) w połączeniu z [TypeScriptem](#), przygoto-

wano również własne [hooki](#) wspomagające typowanie, takie jak `useDispatchTyped` oraz `useSelectorTyped`. Pozwalają one na bezpieczne typowanie akcji oraz selektorów [reduxa](#) bez konieczności powtarzania adnotacji typów w każdym komponencie. Fragmenty tej implementacji przedstawiono na rysunkach [7.8](#) oraz [7.9](#).

```

const store : EnhancedStore<{ account: AccountSliceProp... = configureStore({
  reducer: {
    account: accountSlice.reducer,
    notification: notificationSlice.reducer,
    spotDetails: spotDetailsModalSlice.reducer,
    searchedSpotsListModal: searchedSpotListModalSlice.reducer,
    expandedSpotMediaGallery: expandedSpotMediaGallerySlice.reducer,
    spotFilters: spotFiltersSlice.reducer,
    chats: chatsSlice.reducer,
    map: mapSlice.reducer,
    sidebar: sidebarSlice.reducer,
    searchedSpots: searchedSpotsSlice.reducer,
    social: socialSlice.reducer,
    spotComments: spotCommentSlice.reducer,
    currentViewSpots: currentViewSpotsSlice.reducer,
    currentViewSpotsListModal: currentViewSpotsListModalSlice.reducer,
    currentViewSpotsParams: currentViewSpotParamsSlice.reducer,
    spotWeather: spotWeatherSlice.reducer,
    expandedSpotGalleryMediaList: expandedSpotGalleryMediaListSlice.reducer,
    expandedSpotMediaGalleryModals:
      expandedSpotMediaGalleryModalsSlice.reducer,
    expandedSpotMediaGalleryFullscreenSizeModal:
      expandedSpotMediaGalleryFullscreenSizeSlice.reducer,
    expandedSpotGalleryCurrentMedia:
      expandedSpotGalleryCurrentMediaSlice.reducer,
    spotAddMediaModal: addSpotMediaModalSlice.reducer,
    forum: forumModalSlice.reducer,
    forumReport: forumReportModalSlice.reducer,
  },
});

export default store; Show usages  Mredosz
export type RootState = ReturnType<typeof store.getState>;
export type AppDispatch = typeof store.dispatch;

```

Rysunek 7.8: Konfiguracja sklepu (Redux store)

```

interface AccountSliceProps { Show usages  ⓘ Mredosz +1
  isLoggedIn: boolean;
  username: string;
}

const initialState: AccountSliceProps = {
  isLoggedIn: localStorage.getItem("is_logged_in") === "true",
  username: localStorage.getItem("username") || "",
};

export const accountSlice : Slice<AccountSliceProps, { setLoggedIn(st... = createSlice({ Show usages  ⓘ Mredosz +1
  name: "account",
  initialState,
  reducers: {
    setIsLoggedIn(state : WritableDraft<AccountSliceProps> ) : void {
      localStorage.setItem("is_logged_in", "true");
      state.isLoggedIn = true;
    },
    signOut(state : WritableDraft<AccountSliceProps> ) : void {
      localStorage.removeItem("is_logged_in");
      localStorage.removeItem("username");
      state.isLoggedIn = false;
      state.username = "";
    },
    setUsername(state : WritableDraft<AccountSliceProps> , action: PayloadAction<string>) : void {
      localStorage.setItem("username", action.payload);
      state.username = action.payload;
    },
  },
});

export const accountAction : CaseReducerActions<{ setLoggedIn(state: W... = accountSlice.actions; Show usages  ⓘ Mredosz

```

Rysunek 7.9: Przykładowy slice odpowiedzialny za sprawdzenie czy użytkownik jest zalogowany

### 7.2.3 Integracja i komunikacja z backendem

W niniejszym podrozdziale opisano sposób integracji aplikacji [frontendowej](#) z [backendem](#) oraz mechanizmy odpowiedzialne za bezpieczną i efektywną komunikację z serwerem.

Jest to kluczowy element aplikacji, ponieważ wymaga bezpiecznego przesyłania danych użytkownika. W celu uproszczenia komunikacji z serwerem zdecydowano się na wykorzystanie biblioteki [axios](#) [16] oraz [biblioteki TanStack Query](#) [17]. We



wszystkich ścieżkach wymagających zalogowania użytkownika do zapytania dołączany jest token [JWT](#). Token przekazywany jest w ciasteczku dzięki ustawieniu parametru `withCredentials` na wartość `true`. Przykładem pliku odpowiedzialnego za taką komunikację jest `account.js` (rys. 7.10 i 7.11), który obsługuje operacje związane z logowaniem rejestracją, zmianą hasła oraz wylogowaniem.

```
export async function loginUser(user) { Show usages  Adam Langmesser +1
  return await axios.post(`${BASE_URL}/public/account/login`, user, {
    withCredentials: true,
  });
}

export async function registerUser(user) { Show usages  Mredosz +2
  return await axios.post(`${BASE_URL}/public/account/register`, user, {
    withCredentials: true,
  });
}

export async function sentEmailWithNewPasswordLink(email) { Show usages  Adam Langmesser +1 *
  return await axios.post(
    `${BASE_URL}/public/account/forgot-password`,
    email,
    {
      headers: {
        "Content-Type": "text/plain",
      },
    },
  );
}
```

Rysunek 7.10: Implementacja modułu account (1)

```

export async function changePassword(userData) { Show usages  ⓘ stanoz +1
  return await axios.post(
    `${BASE_URL}/public/account/set-new-password`,
    userData,
  );
}

export async function logout() { Show usages  ⓘ stanoz +1
  await axios.post(
    `${BASE_URL}/account/oauth2/logout`,
    {},
    {
      withCredentials: true,
    },
  );
}

export const googleLoginUrl = `${BASE_URL}/oauth2/authorization/google`; Show usages  ⓘ stanoz
export const githubLoginUrl = `${BASE_URL}/oauth2/authorization/github`; Show usages  ⓘ stanoz

```

Rysunek 7.11: Implementacja modułu `account` (2)

Funkcje odpowiedzialne za komunikację z backendem umieszczono w katalogu `/http`. Dzięki temu są one scentralizowane i mogą być w prosty sposób wykorzystywane w różnych częściach aplikacji. Zastosowanie TanStack Query umożliwiło znaczące ograniczenie powtarzalnego kodu oraz uprościło obsługę błędów i stanów zapytania (takich jak ładowanie danych, błąd czy sukces). Biblioteka udostępnia m.in. wartość `isLoading`, dzięki czemu komponent może łatwo wyświetlić ekran ładowania bez konieczności ręcznego zarządzania własnym stanem. Dodatkowo [hook `useQuery`](#) pozwala na automatyczne pobieranie danych po wejściu na daną podstronę. Komponent deklaruje jedynie, jakie dane są mu potrzebne, a TanStack Query realizuje ich pobranie, cache'owanie oraz odświeżanie. Do operacji wymagających wywołania akcji po stronie użytkownika (np. wysłania formularza logowania) wykorzystywany jest [hook `useMutation`](#) z TanStack Query. Przykład użycia tego rozwiązania w procesie logowania przedstawiono na rys. 7.12.

```

const { mutateAsync, isSuccess, error } = useMutation({
  mutationFn: loginUser,
});

const handleSubmit : (event: FormEvent<HTMLFormElement>) => Pr... = async (event: FormEvent<HTMLFormElement>) : Promise<void> => {
  event.preventDefault();
  await mutateAsync({
    username: enteredValue.username,
    password: enteredValue.password,
  });
  navigate(-1);
};

```

Rysunek 7.12: Wykorzystanie TanStack Query przy logowaniu użytkownika

## 7.2.4 Style

W niniejszym podrozdziale przedstawiono zastosowane w projekcie rozwiązania dotyczące stylowania interfejsu użytkownika oraz narzędzia wykorzystywane do tworzenia spójnej i [responsywnej](#) warstwy wizualnej aplikacji.

Do stylowania interfejsu wykorzystano [framework](#) Tailwind CSS [18]. Dzięki gotowym klasom udostępnianym przez Tailwind wygląd elementów można definiować bezpośrednio w kodzie komponentu, bez konieczności przechodzenia do osobnych plików ze stylami. Ułatwia to zarówno tworzenie widoków, jak i późniejsze modyfikacje — w przypadku zmiany stylu dokładnie wiadomo, gdzie należy jej dokonać. Korzystanie ze zdefiniowanych klas pozwoliło zachować spójność wizualną w całej aplikacji. W pliku `index.css` zdefiniowano zmienne kolorystyczne (rys. 7.13 i 7.14). Dzięki temu zmiana motywu kolorystycznego w przyszłości sprowadza się do edycji wartości w jednym miejscu.

	<code>--height-1\10: 10%;</code>
	<code>--breakpoint-3xl: 160rem;</code>
	<code>--color-mainBlue: #4242f0;</code>
	<code>--color-mainBlueDarker: #0d0db5;</code>
	<code>--color-darkText: #e5e5e5;</code>
	<code>--color-darkBg: #0f0f10;</code>
	<code>--color-darkBgSoft: #1b1c1d;</code>
	<code>--color-grayBg: #d9d9d9;</code>
	<code>--color-darkBgMuted: #323539;</code>
	<code>--color-darkBorder: #939394;</code>
	<code>--color-lightText: #222222;</code>
	<code>--color-lightBg: #e4e3e3;</code>
	<code>--color-lightBgDarker: #cccaca;</code>
	<code>--color-lightBgSoft: #ffffff;</code>
	<code>--color-lightBgMuted: #f2f2f2;</code>
	<code>--color-lightBorder: #fbfdff;</code>
	<code>--color-lightGrayishViolet: #f2eef9;</code>
	<code>--color-whiteSmoke: #f6f6f6;</code>
	<code>--color-warmerWhiteSmoke: #ece9e9;</code>
	<code>--color-lightGrayishBlue: #e5e9ee;</code>
	<code>--color-paleBlueGray: #acafbb;</code>
	<code>--color-grayText: #d3d3d3;</code>

Rysunek 7.13: Implementacja zmiennych kolorystycznych (1)

	<code>--color-violetDark: #363041;</code>
	<code>--color-violetLight: #6d6183;</code>
	<code>--color-violetLightDarker: #4f4660;</code>
	<code>--color-violetLightDark: #554a69;</code>
	<code>--color-violetLighter: #9b8cbd;</code>
	<code>--color-violetDarker: #2c2734;</code>
	<code>--color-violetHeavyDark: #1e1b23;</code>
	<code>--color-violetBtnBorderDark: #625b6e;</code>
	<code>--color-violetBright: #835ace;</code>
	<code>--color-darbVioletBtnOutline: #816ba6;</code>
	<code>--color-mediumDarkBlue: #424b77;</code>
	<code>--color-first: #2c3e50;</code>
	<code>--color-second: #34495e;</code>
	<code>--color-third: #1abc9c;</code>
	<code>--color-fourth: #16a085;</code>
	<code>--color-fifth: #ecf0f1;</code>
	<code>--color-sixth: #e94560;</code>
	<code>--color-magenta: #a01bc1;</code>
	<code>--color-darkYellow: #c5a03c;</code>
	<code>--color-ratingStarColor: #fadb14;</code>
	<code>--color-locationMarkerDarkerBlue: #a3dcff;</code>
	<code>--color-locationMarkerLightBlue: #52bafb;</code>
	<code>--color-userLocationDot: #4285f4;</code>
	<code>--color-spotLocationMarker: #a8071a;</code>

Rysunek 7.14: Implementacja zmiennych kolorystycznych (2)

W niektórych miejscach konieczne było zapisanie stylów w czystym [CSS](#), ponieważ część użytych [bibliotek](#) tego wymagała. W innych przypadkach wystarczyło skorzystać z klas zdefiniowanych w `index.css` oraz klas Tailwinda. Część aplikacji jest [responsywna](#). Tailwind udostępnia predefiniowane prefiksy [responsywne](#) (np. `md:`, `lg:`) (rys. 7.15), utworzono również własny (`3xl:`) na ekrany o rozdzielczości 2560px. Pozwalają one przypisywać style zależnie od szerokości ekranu bez pisania własnych reguł [media queries](#). Dzięki temu implementacja widoków mobilnych i desktopowych była znacząco szybsza.

```
<div className="mt-17 flex flex-col items-center gap-7 lg:mt-0 lg:-ml-40 lg:flex-row xl:-ml-42 xl:gap-10 2xl:-ml-80">
  <div className="relative">
    <img
      alt="profileImage"
      src={userData?.profilePhoto}
      className="dark:drop-shadow-darkBgMuted aspect-square h-64 rounded-full
        shadow-md sm:h-80 lg:h-85 xl:h-96 dark:drop-shadow-md"
    />
```

Rysunek 7.15: Przykładowe użycie klas Tailwind (w tym prefiksów responsywności)

Tailwind został też wykorzystany do obsługi trybu jasnego i ciemnego. Wystarczy dodać klasę z prefiksem `dark:` (np. `dark:bg-black`), aby zmienić kolorystykę elementu, gdy aplikacja jest w trybie ciemnym (rys. 7.16).

```
<input
  id={id}
  value={value}
  type={type}
  onChange={onChange}
  onFocus={setFocusedToTrue}
  onBlur={handleOnBlur}
  className="dark:bg-darkBgMuted bg-lightBgMuted dark:text-darkText text-lightText w-full rounded-md
    p-2 shadow-md focus:outline-none dark:shadow-black/50"
/>
```

Rysunek 7.16: Przykładowe użycie klas Tailwind (w tym wariantu `dark:`)

Aby uzyskać płynniejsze i przyjemniejsze animacje, wykorzystano [bibliotekę Motion](#) [19]. Pozwala ona w prosty sposób tworzyć animacje elementów interfejsu, bez potrzeby ręcznego pisania złożonych reguł [CSS](#). W aplikacji wykorzystano ją

m.in. w polach formularza logowania i rejestracji (rys. 7.17). Na początku etykieta pola (np. „username”) jest wyświetlana wewnątrz pola tekstowego, natomiast po kliknięciu w pole jest płynnie przesuwana nad to pole, co poprawia czytelność i ergonomię formularza.

```
<motion.label
  htmlFor={id}
  initial={false}
  animate={{
    top: shouldFloat ? "-0.7rem" : "0.5rem",
    left: "0.75rem",
    fontSize: shouldFloat ? "0.75rem" : "1rem",
    opacity: shouldFloat ? 1 : 0.6,
  }}
  transition={{ type: "spring", stiffness: 300, damping: 25 }}
  className="dark:text-darkText text-lightText pointer-events-none absolute z-10 px-1 capitalize"
>
  {label}
</motion.label>
<input
  id={id}
  value={value}
  type={type}
  onChange={onChange}
  onFocus={setFocusedToTrue}
  onBlur={handleOnBlur}
  className="dark:bg-darkBgMuted bg-lightBgMuted dark:text-darkText text-lightText w-full rounded-md
p-2 shadow-md focus:outline-none dark:shadow-black/50"
/>
```

Rysunek 7.17: Implementacja animacji z wykorzystaniem Motion

## 7.2.5 Wyszukiwarka spotów

W niniejszym rozdziale przedstawiono sposób implementacji wyszukiwarki spotów.

Jednym z głównych modułów aplikacji jest wyszukiwarka spotów, umożliwia-  
jąca szybkie odnalezienie interesujących lokalizacji. Funkcjonuje ona w dwóch wa-  
riantach: prostym i zaawansowanym (rys. 7.18 oraz 7.19).

```

<div className={`${dark:bg-darkBg} ${dark:text-darkText} ${bg-lightBg} ${text-lightText}
flex min-h-screen w-full flex-col items-center space-y-4 overflow-hidden p-8 pt-18">
  <Switch />
  <SearchBar
    onSetSpots={handleSetSearchedSpots}
    loadMoreRef={loadMoreRef}
    onSetFetchingNextPage={setIsFetchingNextPage}
  />
  <div className="flex w-full flex-col items-center space-y-4">
    <h1 className="text-center text-3xl">The Most Popular Spots</h1>
    <div className="flex w-full flex-col items-center space-y-5">
      <Carousel spots={data!} spotsPerPage={spotsPerPage} />
      <SearchSpotList
        spots={searchedSpots}
        isFetchingNextPage={isFetchingNextPage}
        loadMoreRef={loadMoreRef}
      />
    </div>
  </div>
</div>

```

Rysunek 7.18: Implementacja prostej wersji wyszukiwarki

```

<div className={`${dark:bg-darkBg} ${dark:text-darkText} ${bg-lightBg} ${text-lightText}
flex min-h-screen w-full flex-col items-center space-y-4 overflow-hidden p-8 pt-18">
  <Switch />
  <AdvanceSearchBar
    onSetSpots={handleSetSearchedSpots}
    loadMoreRef={loadMoreRef}
    onSetFetchingNextPage={setIsFetchingNextPage}
  />
  <div className="flex w-full flex-col items-center space-y-10">
    <SearchSpotList
      spots={searchedSpots}
      loadMoreRef={loadMoreRef}
      isFetchingNextPage={isFetchingNextPage}
    />
  </div>
</div>

```

Rysunek 7.19: Implementacja zaawansowanej wersji wyszukiwarki



Przełączanie pomiędzy tymi widokami odbywa się za pomocą przycisku umieszczonego w górnej części strony (rys. 7.20).

```
<div className="dark:shadow-darkBgSoft flex rounded-full shadow-lg shadow-black/20">
  <NavLink
    to="/"
    className={({ isActive } : NavLinkRenderProps ) : string =>
      `dark:shadow-darkBgSoft flex rounded-l-full px-2.5 py-1.5
        transition-all duration-300 ${isActive ? "dark:bg-violetDark bg-violetLight" : ""}`
  >
    Simple filters
  </NavLink>
  <NavLink
    to="/advanced"
    className={({ isActive } : NavLinkRenderProps ) : string =>
      `dark:shadow-darkBgSoft flex rounded-r-full px-2.5 py-1.5
        transition-all duration-300 ${isActive ? "dark:bg-violetDark bg-violetLight" : ""}`
  >
    Advanced filters
  </NavLink>
</div>
```

Rysunek 7.20: Implementacja komponentu do przełączania trybów

W trybie prostym prezentowana jest karuzela (rys. 7.21) z dwunastoma najpopularniejszymi **spotami** w całej aplikacji. W tym widoku możliwe jest wyszukiwanie **spotów** po lokalizacji (kraj, region, miasto).

```

<div className="relative flex w-full items-center justify-center">
  <button
    onClick={() : void => paginate(-1)}
    className="hover:text-darkBorder z-10 cursor-pointer transition-all duration-300"
  >
    <RiArrowLeftWideFill className="text-5xl sm:text-6xl" />
  </button>

  <div className="relative h-[440px] w-full max-w-[1200px] overflow-hidden">
    <AnimatePresence custom={direction} initial={false} mode="sync">
      <motion.div
        key={page}
        custom={direction}
        variants={sliderVariants}
        initial="incoming"
        animate="active"
        exit="exit"
        transition={[ 3 elements... ]}
        className="grid w-full grid-cols-1 grid-rows-1 justify-items-center gap-4
          lg:grid-cols-2 lg:grid-rows-2 2xl:grid-cols-3 2xl:grid-rows-2"
      >
        {currentSpots.map((spot : TopRatedSpot ) : Element => (
          <MostPopularSpot
            spot={spot}
            key={` ${spot.id}-${page}`}
          />
        ))}
      </motion.div>
    </AnimatePresence>
  </div>

  <button
    onClick={() : void => paginate(1)}
    className="hover:text-darkBorder z-10 cursor-pointer transition-all duration-300"
  >
    <RiArrowRightWideFill className="text-5xl sm:text-6xl" />
  </button>
</div>

```

Rysunek 7.21: Implementacja karuzeli z najpopularniejszymi [spotami](#)

Widok zaawansowany udostępnia rozszerzoną wyszukiwarke, która umożliwia filtrowanie wyników po mieście, tagach oraz ocenie, a także ich sortowanie według popularności i średniej oceny (rys. 7.19).

Wyszukiwarka spotów została zbudowana z dwóch głównych komponentów: `HomePage` oraz `AdvanceHomePage`. W skład prostej wersji wchodzi następujące komponenty:

- `Switch` – służy do przełączania widoku między trybem podstawowym a zaawansowanym,
- `SearchBar` – podstawowa wyszukiwarka [spotów](#),
- `Carousel` – wyświetla najpopularniejsze [spoty](#),
- `SearchSpotList` – wyświetla znalezione [spoty](#).

W skład zaawansowanej wersji wchodzi następujące komponenty:

- `Switch` – służy do przełączania widoku między trybem podstawowym a zaawansowanym,
- `AdvanceSearchBar` – zaawansowana wyszukiwarka [spotów](#),
- `SearchSpotList` – wyświetla znalezione [spoty](#).

Komponent `Switch` (rys. 7.20) zawiera dwa elementy `NavLink` z biblioteki `React Router`, co pozwala na przełączanie widoków bez konieczności przeładowywania całej strony.

W komponencie `SearchBar` (rys. 7.22) po wpisaniu co najmniej dwóch znaków wyświetlana jest lista podpowiedzi dla kraju, regionu oraz miasta, w zależności od aktualnie uzupełnianego pola. Po pojawieniu się listy możliwe jest wybranie odpowiedniej lokalizacji, co ułatwia określenie lokalizacji dostępnych [spotów](#).

```

<div className="dark:bg-darkBgSoft light:bg-lightBgSoft flex w-full flex-col items-center justify-between
space-y-3 rounded-md px-3 py-2 shadow-md md:flex-row md:space-y-0 lg:w-3/4 lg:space-x-3 xl:w-1/2
dark:shadow-black">
  <div className="flex w-full flex-col space-y-2">
    <h1>Location</h1>
    <div className="flex w-full flex-col space-y-3 md:flex-row md:space-y-0 md:space-x-2">
      {inputList.map(({ id, label } : { readonly label: "Your Country"; readonl... } : Element ) => (
        <div key={id} className="relative w-full">
          <SearchInput
            label={label}
            id={id}
            value={searchLocation[id] ?? ""}
            onChange={(e : ChangeEvent<HTMLInputElement> ) : void =>
              handleSetLocation(id, e.target.value)
            }
            onFocus={() : void => setActiveInput(id)}
          />
          {activeInput === id && suggestions.length > 0 && (
            <SearchSuggestions
              suggestions={suggestions}
              onClick={handleSuggestionClick}
              id={id}
              onClose={() : void => setActiveInput(null)}
            />
          )}
        </div>
      )}
    </div>
  </div>
  <button
    className="dark:bg-darkBgMuted dark:hover:bg-darkBgMuted/80 light:bg-lightBgMuted
    light:hover:bg-lightBgMuted/80 flex w-full cursor-pointer justify-center rounded-md p-2 md:w-fit"
    onClick={handleSearchSpots}
  >
    <FaSearch />
  </button>
</div>

```

Rysunek 7.22: Implementacja prostej wyszukiwarki

Komponent `SearchSpotList` (rys. 7.23) odpowiada za prezentację wyników wyszukiwania. Został w nim zaimplementowany mechanizm przewijania nieskończonego (*infinite scroll*), który automatycznie pobiera kolejne strony wyników w momencie, gdy użytkownik zbliża się do końca listy. Wykorzystuje on listę komponentów `SpotTile`, a także komponent `LoadingSpinner` oraz komunikat informujący o braku wyników, jeżeli nie zostanie odnaleziony żaden *spot*.

```

<>
<ul className="grid w-full grid-cols-1 place-items-center gap-8 xl:grid-cols-2 2xl:grid-cols-3">
  {spots.map((spot : HomePageSpotDto ) : Element => (
    <SpotTile key={spot.id} spot={spot} />
  ))}
</ul>
<div ref={loadMoreRef} className="h-10" />
{isFetchingNextPage && <LoadingSpinner />}
{spots.length === 0 && (
  <p className="text-center text-2xl">
    Search for spots to see results.
  </p>
)}
</>

```

Rysunek 7.23: Implementacja listy do wyświetlania **spotów**

Komponent **SpotTile** zawiera następujące informacje:

- zdjęcie **spota**,
- miasto, w którym się znajduje,
- nazwę **spota**,
- ocenę oraz liczbę ocen,
- tagi,
- podstawowe informacje pogodowe (temperatura i typ pogody),
- dwa przyciski: jeden prowadzący do widoku szczegółów **spota** oraz drugi informujący, jak daleko znajduje się dany **spot**; po kliknięciu przycisku prezentowana jest lokalizacja **spota** na mapie.

Komponent **AdvanceSearchBar** jest zbliżony wyglądem i strukturą do wersji podstawowej, jednak w polu lokalizacji można podać wyłącznie miasto. Dodatkowo dostępna jest możliwość dodawania tagów z przygotowanej listy. Wyszukiwarka umożliwia także filtrowanie po ocenie oraz sortowanie wyników według oceny i popularności z wykorzystaniem komponentów typu **Dropdown**.

Oba widoki (`HomePage` i `AdvanceHomePage`) współdzielą część komponentów, między innymi `Switch` oraz `SearchSpotList`. Dzięki temu kod odpowiedzialny za wyświetlanie listy wyników jest zdefiniowany w jednym miejscu, a zmiany w sposobie prezentacji [spotów](#) wymagają modyfikacji tylko w komponentach współdzielonych.

#### **7.2.6 Mapa**

#### **7.2.7 Chat**

#### **7.2.8 Forum**

#### **7.2.9 Konto użytkownika**

#### **7.2.10 Panel logowania**

### **7.3 Implementacja CI/CD**

# Rozdział 8

## Testy

8.1 Testy jednostkowe

8.2 Testy integracyjne

8.3 Testy E2E

8.4 Wyniki testów i wnioski

## Rozdział 9

### Prezentacja systemu

9.1 Strona główna

9.2 Strona mapy

9.3 Strona chatu

9.4 Strona forum

9.5 Panel logowania

9.6 Panel konta użytkownika



# Rozdział 10

## Nakład pracy

### 10.1 Ogólny nakład pracy

### 10.2 Indywidualne nakłady pracy

#### 10.2.1 Adam Langmesser

#### 10.2.2 Mateusz Redosz

Na projekt poświęciłem łącznie 324 godziny, z czego 237 przeznaczyłem na prace deweloperskie, 111 na pisanie dokumentacji, 19 godzin na [Review kodu](#), 19 na spotkania dotyczące omówienia dalszych prac projektowych oraz przy pomocy innym członkom zespołu oraz 49 godzin poświęciłem nad stworzeniem widoków na figmie. Prace nad częścią deweloperską rozpocząłem 04.08.2024 a zakończyłem 08.09.2025. W projekcie pracowałem nad Rejestracją użytkownika, tokenem [JWT](#), częściową implementacją [CI/CD](#), stroną główną, zaimplementowaniem [Sidebara](#) oraz podstroną dla użytkownika. Moje wylistowane zadania z Jira:

#### 1. Dokumentacja

- TODO

#### 2. [Design](#)

- Ustalić paletę kolorystyczną

- Propozycja wyglądu

### 3. [Backend](#) i [Frontend](#)

- Formularz rejestracji
- Routing
- Formatowanie w React (prettier)
- Obsługa JWT na frontend
- OAuth Frontend
- Update JWT
- Refactor JWT
- Stworzenie komponentu Notification i poprawa błędów
- Implementacja pierwszych testów
- Zaimplementowanie kolejki w komponencie notification
- Dodanie reduxa do rejestracji
- Zmiana sposobu pobierania danych o spotach
- Obsługa customowych błędów z jakarta.validation
- Obsługa auto wylogowania przy starcie
- Domyślna wiadomość w notification
- Poprawa headera
- Ciemny motyw
- Refactor pogody
- Propozycja wyglądu
- Przeniesienie zdjęć z google drive
- Dodać Type script do Reacta
- Aktualizacja tailwinda i dodanie kolorów
- Podstawowy [Sidebar](#)

- Strona główna z prostymi filtrami
- Strona główna z zaawansowanymi filtrami
- [Sidebar](#)
- Strona profilu
- Ustawienia
- Listy spotów
- Lista zdjęć
- Lista filmów
- Lista znajomych
- Dodanie spotów
- Lista komentarzy
- Strona główna profilu
- Listy
- Poprawa [Sidebara](#)
- Zmiana kropki na przyciemnienie tła na [Sidebar](#)
- Poprawa strony do logowania i rejestracji
- Usunięcie username z account Redux
- Dodanie zamknięcia [Sidebara](#) na małych ekranach po kliknięciu nav linka
- Poprawić tooltipa na sidebar
- Zmiana sposobu pobierania username na backendzie z tokena jwt
- Paginacja z infinity scrollem
- Lista zdjęć innego usera
- Walidacja i responsywność w dodaniu spotów
- Dodanie sortowania i filtrów na zaawansowanej stronie
- Zmiana na infinity scrola

- Zmiana zdjęcia profilowego użytkownika
- Czyszczenie formularza w dodawaniu spota
- Dodanie wyszukiwarki znajomych w Social
- Zatwierdzenie przez drugiego użytkownika dodania do znajomych
- Sprawdzenie czy wszystko działa i poprawki Mateusz

#### 4. [CI/CD](#)

- Dodanie testów z frontendu do github actions
- Poprawa github actions
- Poprawa pipeline od Javy i Reacta

#### 5. Praca dyplomowa

- Uzupełnienie informacji o zespole i podział na rozdziały

### 10.2.3 Stanisław Oziemczuk

### 10.2.4 Kacper Badek

# Rozdział 11

## Podsumowanie

11.1 Osiągnięte rezultaty

11.2 Napotkane wyzwania

11.3 Plany na przyszłość

# Rozdział 12

## Słownik pojęć i skrótów

### API

(ang. *application programming interface*); zbiór reguł i operacji do komunikacji z oprogramowaniem.. [16](#), [18](#)

### Backend

Część aplikacji odpowiedzialna za logikę biznesową, przetwarzanie danych i komunikację z bazą danych. Działa po stronie serwera i obsługuje żądania wysyłane przez frontend. [2](#), [15](#), [48](#), [49](#), [51](#), [54](#), [110](#), [128](#)

### Backlog

Lista zadań, które należy wykonać w ramach projektu, używane w metodykach zwinnych.. [15](#)

### Baza danych

Zbiór uporządkowanych danych przechowywanych w sposób umożliwiający ich łatwe wyszukiwanie, modyfikowanie i analizowanie. W aplikacjach najczęściej wykorzystywane są relacyjne lub nierelacyjne bazy danych. [48](#), [49](#), [51](#)

### Biblioteka

Zewnętrzny lub wewnętrzny zestaw gotowych funkcji, klas, komponentów lub modułów, który można wielokrotnie wykorzystywać w projekcie zamiast pisać wszystko od zera. [49](#), [105](#), [107](#), [110](#), [116](#), [121](#)

## BPMN

(ang. *Business Process Model and Notation*); standardowa notacja graficzna, która umożliwia szczegółowe przedstawienie i dokumentowanie procesów biznesowych.. [17](#)

## Cache

Mechanizm przechowywania danych w celu przyspieszenia ich ponownego odczytu. [48](#), [49](#), [136](#)

## CI/CD

Skrót od *Continuous Integration/Continuous Deployment*. Praktyka programistyczna polegająca na automatyzacji procesu budowania, testowania i wdrażania oprogramowania. [16](#), [127](#), [130](#)

## CSS

Kaskadowe arkusze stylów (Cascading Style Sheets) — język opisu prezentacji dokumentów (np. HTML). Definiuje wygląd interfejsu: układ, kolory, typografię, odstępy, animacje i zachowania responsywne, oddzielając warstwę treści od warstwy prezentacji.. [116](#)

## Design

Etap lub proces projektowania wyglądu i funkcjonalności aplikacji, obejmujący zarówno aspekty wizualne, jak i użytkowe (UX/UI). [127](#)

## Disciplined Agile Delivery - Lean Life Cycle

Disciplined Agile Delivery w wariancie Lean Life Cycle to sposób prowadzenia projektu, który łączy elastyczność Agile z przewidywalnością Waterfalla, ale bez stałych sprintów — praca toczy się w ciągłym przepływie. Na starcie zakłada mocniejszą fazę przygotowawczą: doprecyzowanie zakresu, szkic architektury, identyfikację ryzyk i kryteria jakości. W realizacji następuje ciągle doprecyzowywanie wymagań i backlogu, oparte na regularnym feedbacku

udziałowców. Całość opiera się na praktykach Lean oraz lekkim governance: code review i regularnych przeglądach postępów. . [10](#)

## **Droniarz**

Potoczne określenie osoby, która jest jednocześnie pilotem oraz operatorem drona. Zwykle entuzjasta dronów.. [8](#), [9](#), [140](#)

## **Droniarz foto/video**

Pilot wykorzystujący drony fotograficzne/filmowe do rejestracji materiałów wizualnych (zdjęcia, wideo), zwykle z naciskiem na stabilizację i jakość obrazu.. [17](#)

## **Folder by type**

Sposób organizowania struktury katalogów w projekcie, w którym pliki są grupowane według rodzaju (typu) zasobu, a nie według funkcjonalności. Na przykład wszystkie komponenty trafiają do jednego folderu, wszystkie style do innego itd. [54](#), [102](#)

## **Framework**

Zestaw narzędzi, bibliotek i struktur wspomagających tworzenie aplikacji. Ułatwia programowanie poprzez dostarczenie gotowych komponentów oraz określenie zasad organizacji kodu. [2](#), [49](#), [54](#), [113](#)

## **Frontend**

Warstwa aplikacji odpowiedzialna za interfejs użytkownika oraz interakcję z użytkownikiem. Zazwyczaj tworzona przy użyciu technologii takich jak HTML, CSS i JavaScript. [2](#), [15](#), [48](#), [49](#), [51](#), [102](#), [110](#), [128](#)

## **Hook (React)**

Prosta funkcja w React, która „dodaje” możliwości do elementu interfejsu — np. pozwala mu coś zapamiętać (stan) albo zrobić coś po zmianie/załadowaniu. Wszystkie hooki zaczynają się od `use...` (np. `useState`, `useEffect`).. [107](#), [108](#), [112](#)



## IDE

(ang. *integrated development environment*); to zintegrowane środowisko programistyczne, służące do tworzenia, modyfikowania, testowania i konserwacji oprogramowania. [14](#)

## Infinite scroll

Wzorzec interfejsu użytkownika, w którym kolejne porcje treści są automatycznie doładowywane podczas przewijania strony w dół, zamiast być podzielone na odrębne, ręcznie przełączane strony. [122](#)

## JWT

Skrót od *JSON Web Token*. Standard służący do bezpiecznego przekazywania informacji między stronami w formacie JSON, często używany w procesach autoryzacji użytkowników. [111](#), [127](#)

## Media queries

Konstrukcja CSS pozwalająca stosować reguły stylów w zależności od cech urządzenia/okna (np. szerokości ekranu, orientacji, preferencji użytkownika). Podstawa responsywnego projektowania (*responsive design*).. [116](#), [137](#)

## PANSA

Polish Air Navigation Services Agency, pol. Polska Agencja Żeglugi Powietrznej. Instytucja ta zapewnia m.in. mapę z zaznaczonymi strefami lotów. Każda strefa ma swoje właściwości prawne. . [21](#)

## Parametry zapytania (query params)

Pary **klucz=wartość** przekazywane w części adresu URL po znaku zapytania **?**, służące m.in. do filtrowania, sortowania, paginacji wyników lub przekazywania dodatkowych opcji żądania do serwera. [57](#), [61](#), [62](#), [64–75](#), [77–79](#), [81–84](#), [86–94](#), [96–99](#)

## Props

Właściwości przekazywane do komponentu React przez komponent nadrzędny; służą do konfiguracji i przekazywania danych. Powinny być traktowane jako tylko do odczytu (read-only) wewnątrz komponentu potomnego.. [107](#)

## Protected route

Trasa w aplikacji, do której dostęp jest ograniczony, zwykle tylko dla zalogowanych użytkowników lub użytkowników z odpowiednimi uprawnieniami. Jeżeli użytkownik nie spełnia warunków, jest przekierowywany (np. na stronę główną). [106](#)

## React

Biblioteka JavaScript do budowy interfejsów użytkownika w oparciu o komponenty deklaratywne i wirtualny DOM. Zapewnia jednokierunkowy przepływ danych oraz zarządzanie stanem komponentów.. [48](#), [49](#), [107](#)

## Redis

Baza danych typu klucz–wartość wykorzystywana jako szybka warstwa [cache](#). [48](#), [49](#), [51](#)

## Redux

Biblioteka do przewidywalnego zarządzania stanem aplikacji. Opiera się na jednym *store*, akcjach i czystych *reducerach*, promuje niemutowalność i jednokierunkowy przepływ danych. Często używana z Reactem, ale niezależna od niego.. [107](#), [108](#)

## Responsywność

Określenie związane z projektowaniem responsywnym (Responsive Web Design, RWD), czyli dostosowywaniem interfejsu do różnych rozmiarów i parametrów ekranów. Obejmuje m.in. elastyczne siatki, grafiki i [Media queries](#), tak aby układ i czytelność były zachowane na telefonach, tabletach i desktopach.. [113](#), [116](#)

## REST API

Architektura budowania usług sieciowych komunikujących się poprzez metody protokołu HTTP (GET, PUT, POST, DELETE, PATCH). Wymiana danych występuje często w formacie JSON lub XML.

REST API musi spełniać następujące reguły:

1. **Rozdzielenie klient-serwer** — klient i serwer są od siebie niezależne, komunikują się poprzez interfejs.
2. **Bezstanowość** — każde żądanie przez klienta zawiera wszystkie informacje niezbędne do jego obsłużenia. Po otrzymaniu żądania serwer nie przechowuje o nim żadnych informacji.
3. **Buforowalność (cache)** — odpowiedzi z API powinny informować, czy dane można cache'ować. Jeśli tak, to przy kolejnym żądaniu mogą być zwrócone z cache'a.
4. **Jednolity interfejs:**
  - **Identyfikacja zasobów** — każdy zasób musi być jednoznacznie zidentyfikowany w interakcji klient-serwer.
  - **Manipulacja zasobów poprzez reprezentację** — po otrzymaniu reprezentacji klient może zmienić stan zasobu przesyłając zmodyfikowaną reprezentację.
  - **Samoopisujące się wiadomości** — każde żądanie i odpowiedź powinny zawierać informacje do jego poprawnego przetworzenia.
  - **Hypermedia jako silnik stanu aplikacji (HATEOAS)** — po otrzymaniu odpowiedzi klient powinien móc dynamicznie poznać inne interakcje przez linki.
5. **Warstwowość** — klient nie wie czy komunikuje się bezpośrednio z serwerem, czy poprzez pośrednika (np. proxy) oraz nie wie z czym komunikuje się obsługująca go warstwa.
6. **Kod na żądanie (opcjonalnie)** — serwer może przesłać fragment kodu, który zostanie wykonany przez klienta.

[15](#), [49](#), [54](#), [57](#)

## Review kodu

Proces polegający na wzajemnym przeglądzie kodu źródłowego przez programistów w celu wykrycia błędów, poprawy jakości oraz zwiększenia spójności projektu. [15](#), [16](#), [127](#)

## Sidebar

Boczny panel w interfejsie użytkownika, zawierający menu nawigacyjne lub dodatkowe opcje funkcjonalne aplikacji. [47](#), [127–129](#)

## Spot

Spotkanie zespołu projektowego, zazwyczaj krótkie i regularne, służące omówieniu postępów prac, problemów oraz planów na najbliższy okres. [15](#), [119–124](#)

## Stan

Aktualny zestaw danych przechowywanych przez aplikację lub komponent, na podstawie którego renderowany jest interfejs użytkownika. Stan może być lokalny (utrzymywany w pojedynczym komponencie) albo globalny (wspólny dla wielu komponentów).. [107](#)

## Tablica Kanban

Narzędzie do zarządzania przepływem pracy, które pomaga zespołom śledzić zadania oraz ich postępy. Składa się z kolumn reprezentujących stan etapu prac, na przykład „Do zrobienia” lub „W trakcie”.. [15](#)

## TypeScript

Rozszerzenie do języka JavaScript dodający statyczne typowanie, interfejsy i narzędzia do większych projektów. Kompiluje się do czystego JavaScript, ułatwiając wykrywanie błędów w czasie kompilacji i refaktoryzację.. [48](#), [49](#), [107](#)

## UI

Interfejs użytkownika (ang. *User Interface*); warstwa prezentacji odpowiedzialna za sposób wyświetlania danych oraz interakcji użytkownika z aplikacją.. [16](#), [107](#)

## UML

(ang. *Unified Modeling Language*); graficzny język wizualizacji, specyfikowania oraz dokumentowania składników systemów informatycznych. . [17](#)

# Spis tabel

Tabela 2.1: Karta udziałowca: Zespół projektowy . . . . .	7
Tabela 2.2: Karta udziałowca: Promotor . . . . .	8
Tabela 2.3: Karta udziałowca: Droniarze . . . . .	9
Tabela 3.1: Usługa zewnętrzna: GitHub Actions (CI) . . . . .	18
Tabela 3.2: Usługa zewnętrzna: Azure Blob Storage . . . . .	18
Tabela 3.3: Usługa zewnętrzna: Mailtrap . . . . .	18
Tabela 3.4: Usługa zewnętrzna: LocationIQ . . . . .	19
Tabela 3.5: Usługa zewnętrzna: Google Maps (Maps URLs) . . . . .	19
Tabela 3.6: Usługa zewnętrzna: OpenFreeMap . . . . .	19
Tabela 3.7: Usługa zewnętrzna: Open-Meteo . . . . .	20
Tabela 3.8: Usługa zewnętrzna: Tenor GIF API . . . . .	20
Tabela 3.9: Usługa zewnętrzna: Where the ISS at? . . . . .	20
4.1 Profil użytkownika . . . . .	32
4.2 Lista dodanych spotów . . . . .	33
4.3 Dodanie spota . . . . .	34
4.4 Lista zdjęć . . . . .	35
4.5 Lista filmów . . . . .	35
4.6 Lista znajomych . . . . .	36
4.7 Lista obserwujących . . . . .	36
4.8 Lista obserwowanych . . . . .	37
4.9 Lista polubionych/odwiedzonych/planowanych spotów . . . . .	37
4.10 Lista komentarzy . . . . .	38
4.11 Ustawienia profilu . . . . .	39

4.12	Resetowanie hasła . . . . .	40
4.13	Dodawanie do znajomych . . . . .	41
4.14	Logowanie i rejestracja . . . . .	42
4.15	Strona główna — podstawowe filtry . . . . .	43
4.16	Strona główna — zaawansowane filtry . . . . .	44
4.17	Ustawienia motywu (ręczna zmiana) . . . . .	45
4.18	Zapamiętanie preferencji motywu . . . . .	46
4.19	Szybki przełącznik motywu w interfejsie . . . . .	47
Tabela 7.1: Zestawienie wszystkich endpointów HTTP systemu . . . . .		61
Tabela 7.2: Karta endpointu: /public/user-dashboard/profile/{targetUsername} . . . . .		62
Tabela 7.3: Karta endpointu: /user-dashboard/favorite-spots . . . . .		63
Tabela 7.4: Karta endpointu: /user-dashboard/photos . . . . .		64
Tabela 7.5: Karta endpointu: /user-dashboard/add-spot . . . . .		65
Tabela 7.6: Karta endpointu: /user-dashboard/add-spot . . . . .		66
Tabela 7.7: Karta endpointu: /public/spot/gallery . . . . .		67
Tabela 7.8: Karta endpointu: /public/spot/current-view . . . . .		68
Tabela 7.9: Karta endpointu: /public/spot/get-spot-basic-weather . . . . .		69
Tabela 7.10: Karta endpointu: /public/spot/get-spot-detailed-weather . . . . .		70
Tabela 7.11: Karta endpointu: /public/spot/get-spot-wind-speeds . . . . .		71
Tabela 7.12: Karta endpointu: /public/spot/most-popular . . . . .		72
Tabela 7.13: Karta endpointu: /public/spot/search/home-page . . . . .		73
Tabela 7.14: Karta endpointu: /public/spot/search/home-page/locations . . . . .		74
Tabela 7.15: Karta endpointu: /public/spot/search/home-page/advance . . . . .		75
Tabela 7.16: Karta endpointu: /public/spot/{spotId}/comments . . . . .		76
Tabela 7.17: Karta endpointu: /public/spot/{spotId}/comments/{commentId} . . . . .		77
Tabela 7.18: Karta endpointu: /spot/{spotId}/comments . . . . .		78
Tabela 7.19: Karta endpointu: /spot/comments/{commentId}/vote . . . . .		78
Tabela 7.20: Karta endpointu: /spot/comments/vote-type . . . . .		79
Tabela 7.21: Karta endpointu: /public/post/{postId} . . . . .		80
Tabela 7.22: Karta endpointu: /post . . . . .		81
Tabela 7.23: Karta endpointu: /post/{postId} . . . . .		82

Tabela 7.24: Karta endpointu: /post/{postId}/vote . . . . .	82
Tabela 7.25: Karta endpointu: /public/categories-tags . . . . .	83
Tabela 7.26: Karta endpointu: /public/post/{postId}/comments . . . . .	85
Tabela 7.27: Karta endpointu: /post/{postId}/comments . . . . .	86
Tabela 7.28: Karta endpointu: /post/comments/{commentId} . . . . .	87
Tabela 7.29: Karta endpointu: /post/comments/{commentId}/vote . . . . .	87
Tabela 7.30: Karta endpointu: /comments/{commentId}/replies . . . . .	88
Tabela 7.31: Karta endpointu: /public/account/register . . . . .	89
Tabela 7.32: Karta endpointu: /public/account/login . . . . .	89
Tabela 7.33: Karta endpointu: /public/account/forgot-password . . . . .	90
Tabela 7.34: Karta endpointu: /public/account/set-new-password . . . . .	91
Tabela 7.35: Karta endpointu: /account/check . . . . .	91
Tabela 7.36: Karta endpointu: /public/upload/media . . . . .	92
Tabela 7.37: Karta endpointu: /gifs/trending . . . . .	93
Tabela 7.38: Karta endpointu: /gifs/search . . . . .	94
Tabela 7.39: Karta endpointu: /chats/{chatId}/messages . . . . .	95
Tabela 7.40: Karta endpointu: /chats/get-or-create-private-chat . . . . .	96
Tabela 7.41: Karta endpointu: /chats/{chatId}/send-files . . . . .	97
Tabela 7.42: Karta endpointu: /chats/create/group . . . . .	98
Tabela 7.43: Karta endpointu: /chats/{chatId} . . . . .	99
Tabela 7.44: Karta endpointu: /chats/group-chat/add/search/{chatId} . . . . .	100



# Bibliografia

- [1] *Disciplined Agile Delivery*. PMI. 1 sty. 2025. URL: <https://www.pmi.org/disciplined-agile/process/introduction-to-dad/why> (term. wiz. 30. 10. 2025).
- [2] *Disciplined Agile Delivery — Lean Life Cycle*. PMI. 1 sty. 2025. URL: <https://www.pmi.org/disciplined-agile/lifecycle/lean-lifecycle> (term. wiz. 30. 10. 2025).
- [3] Stanisław Wrycza, Bartosz Marcinkowski i Krzysztof Wyrzykowski. „Język UML 2.0 w modelowaniu systemów informatycznych”. Warszawa: Helion, 2006. ISBN: 83-736-1892-9, 8373618929.
- [4] Michał Wolski. *10 wskazówek poprawiających modelowanie procesów biznesowych w notacji BPMN*. 14 maj. 2024. URL: <https://wolski.pro/2024/05/10-wskazowek-poprawiajacych-modelowanie-procesow-biznesowych-w-notacji-bpmn/> (term. wiz. 19. 11. 2025).
- [5] *About billing for GitHub Actions*. GitHub Docs. 1 sty. 2024. URL: <https://docs.github.com/en/billing/managing-billing-for-github-actions/about-billing-for-github-actions> (term. wiz. 02. 11. 2025).
- [6] *Scalability and performance targets for Blob storage*. Microsoft Learn. 1 sty. 2024. URL: <https://learn.microsoft.com/azure/storage/blobs/scalability-targets> (term. wiz. 02. 11. 2025).
- [7] *What are the limitations in Mailtrap?* Mailtrap Docs. 1 sty. 2024. URL: <https://help.mailtrap.io/article/111-what-are-the-limitations-in-mailtrap/> (term. wiz. 02. 11. 2025).
- [8] *LocationIQ Pricing*. LocationIQ. 1 sty. 2024. URL: <https://locationiq.com/pricing> (term. wiz. 02. 11. 2025).
- [9] *Google Maps (Maps URLs)*. Google Maps. 1 sty. 2024. URL: <https://developers.google.com/maps/documentation/urls/get-started?hl=pl> (term. wiz. 02. 11. 2025).
- [10] *OpenFreeMap Documentation*. OpenFreeMap. 1 sty. 2024. URL: <https://openfreemap.org/docs> (term. wiz. 02. 11. 2025).

- [11] *Open-Meteo API Usage & Pricing*. Open-Meteo. 1 sty. 2024. URL: <https://open-meteo.com/en/docs/usage-and-pricing> (term. wiz. 02.11.2025).
- [12] *Tenor API — Documentation*. Tenor. 1 sty. 2024. URL: <https://tenor.com/gifapi/documentation> (term. wiz. 02.11.2025).
- [13] *Where the ISS at? API*. wheretheiss.at. 1 sty. 2024. URL: <https://wheretheiss.at/> (term. wiz. 02.11.2025).
- [14] *React useState*. 1 sty. 2025. URL: <https://react.dev/reference/react/useState> (term. wiz. 03.11.2025).
- [15] *Redux*. 1 sty. 2025. URL: <https://redux.js.org/> (term. wiz. 03.11.2025).
- [16] *Axios*. 1 sty. 2025. URL: <https://axios-http.com/> (term. wiz. 03.11.2025).
- [17] *Tanstack Query*. 1 sty. 2025. URL: <https://tanstack.com/query/latest> (term. wiz. 03.11.2025).
- [18] *Tailwind*. 1 sty. 2025. URL: <https://tailwindcss.com/> (term. wiz. 03.11.2025).
- [19] *Motion*. 1 sty. 2025. URL: <https://motion.dev/> (term. wiz. 03.11.2025).

# Załączniki

Płyta CD z następującą zawartością:

- *pliki projektowe* – pliki składające się na całość projektu
  - repozytorium kodu źródłowego wraz z instrukcją zbudowania i uruchomienia projektu
  - źródło pracy inżynierskiej.
- *Langmesser Adam\_Redosz Mateusz\_Oziemczuk Stanisław\_Badek Kacper\_praca pisemna* – katalog zawierający plik PDF z pracą inżynierską.