



# 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> 7 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 . . . . .	8
2.4	Wizja rozwiązania . . . . .	8
2.5	Aspekty społeczne i biznesowe . . . . .	8
2.5.1	Aspekty społeczne . . . . .	8
2.5.2	Aspekty biznesowe . . . . .	8
<b>3</b>	<b>Planowanie</b>	<b>9</b>
3.1	Metodologia pracy . . . . .	9
3.1.1	Przegląd rozważanych podejść . . . . .	9
3.1.2	Odrzucone podejścia . . . . .	9
3.1.3	Wybrane podejście: Disciplined Agile Delivery (Lean Life Cycle) . . . . .	10
3.1.4	Narzędzia i komunikacja . . . . .	10
3.1.5	Podział ról w zespole . . . . .	11
3.2	Harmonogram projektu . . . . .	11
3.3	Technologie i narzędzia . . . . .	13
3.3.1	Technologie . . . . .	13

3.3.2	Narzędzia . . . . .	13
3.4	Zasoby i ograniczenia . . . . .	16
3.4.1	Zasoby . . . . .	16
3.4.2	Ograniczenia . . . . .	16
3.4.3	Usługi zewnętrzne . . . . .	16
3.5	Analiza ryzyka . . . . .	19
<b>4</b>	<b>Analiza wymagań</b>	<b>20</b>
4.1	Przypadki użycia . . . . .	21
4.1.1	Aktorzy . . . . .	21
4.1.2	Diagram przypadków użycia . . . . .	22
4.1.3	Scenariusz przypadków użycia . . . . .	22
4.2	Wymagania . . . . .	22
4.2.1	Wymagania ogólne . . . . .	23
4.2.1.1	Wymagania ogólne dla czatu . . . . .	23
4.2.2	Wymagania funkcjonalne . . . . .	25
4.2.2.1	Funkcjonalności dla mapy . . . . .	25
4.2.2.2	Wymagania funkcjonalne dla czatu . . . . .	25
4.2.2.3	Funkcjonalności dla forum . . . . .	39
4.2.2.4	Funkcjonalności dla konta użytkownika . . . . .	39
4.2.2.5	Funkcjonalności dla logowania i rejestracji . . . . .	49
4.2.2.6	Funkcjonalności dla wyszukiwarki spotów . . . . .	50
4.2.2.7	Funkcjonalności dla motywu . . . . .	52
4.2.3	Wymagania pozafunkcjonalne . . . . .	54
4.2.3.1	Wymagania pozafunkcjonalne dla czatu . . . . .	54
<b>5</b>	<b>Projekt</b>	<b>65</b>
5.1	Wzorce projektowe . . . . .	65
5.2	Architektura systemu . . . . .	65
5.2.1	Diagram architektury . . . . .	65
5.2.2	Komponenty systemu . . . . .	65
5.3	Projekt bazy danych . . . . .	65

5.3.1	Model danych . . . . .	65
5.3.2	Diagram ERD . . . . .	65
5.4	Architektura interfejsu użytkownika . . . . .	65
5.4.1	Projekt strony głównej . . . . .	65
5.4.2	Projekt panelu logowania . . . . .	65
5.4.3	Projekt mapy . . . . .	65
5.4.4	Projekt chatu . . . . .	65
5.4.5	Projekt forum . . . . .	65
5.4.6	Projekt konta użytkownika . . . . .	65
<b>6</b>	<b>Przebieg realizacji projektu</b>	<b>66</b>
6.1	Sprint 1 . . . . .	66
6.2	Sprint 2 . . . . .	66
<b>7</b>	<b>Realizacja Projektu</b>	<b>67</b>
7.1	Implementacja backendu . . . . .	67
7.1.1	Struktura projektu . . . . .	67
7.1.2	Integracja z bazą danych . . . . .	67
7.1.3	Obsługa uwierzytelnienia . . . . .	67
7.1.4	Konteneryzacja . . . . .	67
7.2	Implementacja frontendu . . . . .	67
7.2.1	Struktura aplikacji . . . . .	67
7.2.2	Zarządzanie stanem i przepływ danych . . . . .	72
7.2.3	Integracja i komunikacja z backendem . . . . .	74
7.2.4	Style . . . . .	77
7.2.5	Wyszukiwarka spotów . . . . .	81
7.2.6	Mapa . . . . .	88
7.2.7	Chat . . . . .	88
7.2.8	Forum . . . . .	88
7.2.9	Konto użytkownika . . . . .	88
7.2.10	Panel logowania . . . . .	88
7.3	Implementacja CI/CD . . . . .	88

<b>8 Testy</b>	<b>89</b>
8.1 Testy jednostkowe . . . . .	89
8.2 Testy integracyjne . . . . .	89
8.3 Testy E2E . . . . .	89
8.4 Wyniki testów i wnioski . . . . .	89
<b>9 Prezentacja systemu</b>	<b>90</b>
9.1 Strona główna . . . . .	90
9.2 Strona mapy . . . . .	90
9.3 Strona chatu . . . . .	90
9.4 Strona forum . . . . .	90
9.5 Panel logowania . . . . .	90
9.6 Panel konta użytkownika . . . . .	90
<b>10 Nakład pracy</b>	<b>91</b>
10.1 Ogólny nakład pracy . . . . .	91
10.2 Indywidualne nakłady pracy . . . . .	91
10.2.1 Adam Langmesser . . . . .	91
10.2.2 Mateusz Redosz . . . . .	91
10.2.3 Stanisław Oziemczuk . . . . .	94
10.2.4 Kacper Badek . . . . .	94
<b>11 Podsumowanie</b>	<b>95</b>
11.1 Osiągnięte rezultaty . . . . .	95
11.2 Napotkane wyzwania . . . . .	95
11.3 Plany na przyszłość . . . . .	95
<b>12 Słownik pojęć i skrótów</b>	<b>96</b>
<b>Spis tabel</b>	<b>103</b>
<b>Bibliografia</b>	<b>106</b>
<b>Załączniki</b>	<b>108</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	
Identyfikator:	UO1
Nazwa:	Zespół projektowy
Opis:	Zespół czterech studentów odpowiedzialnych za analizę, projekt, implementację, testy oraz dokumentację systemu.
Typ udziałowca:	ożywiony, bezpośredni
Punkt widzenia:	techniczna, wykonawcza
Ograniczenia:	Ograniczone zasoby czasowe i doświadczenie komercyjne.
Wymagania:	Wymagania funkcjonalne i techniczne systemu, możliwość realizacji w ramach projektu dyplomowego.

Tabela 2.1: Zespół projektowy

KARTA UDZIAŁOWCA	
Identyfikator:	UO2
Nazwa:	Promotor
Opis:	Osoba nadzorująca przebieg projektu, weryfikująca poprawność merytoryczną i zgodność z wymaganiami uczelni.
Typ udziałowca:	ożywiony, pośredni
Punkt widzenia:	merytoryczna, formalna, jakościowa
Ograniczenia:	Nie odpowiada za implementację; rekomenduje, opiniuje i zatwierdza.
Wymagania:	Czytelna dokumentacja, zgodność z wytycznymi kierunku i dobry poziom techniczny rozwiązania.

Tabela 2.2: Promotor

KARTA UDZIAŁOWCA	
Identyfikator:	UO3
Nazwa:	<a href="#">Droniarze</a>
Opis:	Główna grupa docelowa systemu – osoby latające dronami rekreacyjnie lub półprofesjonalnie, szukające miejsc do lotów i wymiany doświadczeń.
Typ udziałowca:	ożywiony, bezpośredni
Punkt widzenia:	użytkownik końcowy: prostota obsługi, rzetelne informacje o spotach, wygodne dzielenie się treściami.
Ograniczenia:	Brak wpływu na architekturę techniczną systemu; oczekują intuicyjnego interfejsu.
Wymagania:	Lista spotów, informacje o ograniczeniach prawnych, oceny/komentarze, dodawanie treści, podstawowe funkcje społecznościowe.

Tabela 2.3: 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 wyma-

gano 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 wariacie **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 mieliśmy bezpłatny dostęp za pośrednictwem uczelni, w innych mogliśmy założyć konta edukacyjne, które oferowały dostęp do wszystkich funkcji narzędzia. Gdy żadna z wymienionych opcji nie była udostępniona, wybieraliśmy 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. Używaliśmy 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. Wykorzystywaliśmy je do lokalnego uruchamiania bazy danych oraz serwisu do cachowania.

- **One Drive**

Usługa dysku chmurowego oferowana przez firmę Microsoft. Przechowywaliśmy 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. Wykorzystaliśmy go 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 metodykach zwinnych. Do [Backlogu](#) wpisywaliśmy zadania, a na [tablicy Kanbanowej](#) rejestrowaliśmy ich statusy oraz poświęcony czas.

- **GitHub**

Zdalne repozytorium służące do przechowywania i wersjonowania kodu aplikacji. Zamieściliśmy tam kod naszego projektu. Do każdego zadania tworzyliśmy osobną gałąź z właściwą nazwą, a po zakończeniu prac przeprowadzaliśmy [review kodu](#). Następnie łączyliśmy 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. Stosowaliśmy 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. Wykorzystywaliśmy go podczas [review codu](#). Copilot skanował wszystkie pliki i w komentarzach opisywał sugerowane zmiany lub potencjalne błędy.

- **Discord**

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

- **Messenger**

Komunikator będący usługą Facebooka. Daje możliwość tworzenia czatów grupowych lub prywatnych, a także udostępniania plików. Używaliśmy 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. Wykorzystywaliśmy go do testowania stworzonych endpointów oraz debugowania.

- **Figma**

Narzędzie chmurowe do projektowania interfejsów użytkownika ([UI](#)). Umożliwia zespołowe tworzenie w pełni interaktywnych prototypów. Wykonaliśmy 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]). Zrobiliśmy w nim diagram przypadków użycia.

## 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

<b>Usługa</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)

<b>Usługa</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

<b>Usługa</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

<b>Usługa</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

<b>Usługa</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)

<b>Usługa</b>	OpenFreeMap [10, 11]
<b>Opis</b>	Publiczny serwer kafelków do renderu mapy na froncie.
<b>Limit</b>	30 000 zapytań/mies.

**Tabela 3.6:** Usługa zewnętrzna: OpenFreeMap

<b>Usługa</b>	Open-Meteo [12]
<b>Opis</b>	Prognozy pogody wyświetlane dla spotów.
<b>Limit</b>	10 000 zapytań/dzień

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

<b>Usługa</b>	Tenor GIF API [13]
<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

<b>Usługa</b>	Where the ISS at? [14]
---------------	------------------------

<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

Niniejszy rozdział zawiera analizę wymagań postawionych systemowi.

Do określenia priorytetów realizacji wymagań skorzystano z metody MoSCoW. Metoda MoSCoW jest techniką priorytetyzacji wymagań. Polega ona na przypisaniu każdemu wymaganiu jednej z czterech kategorii priorytetu, określających jego znaczenie dla minimalnie użytecznej wersji systemu.

W niniejszej pracy przyjęto następującą interpretację priorytetów MoSCoW:

**M – *Must have*** wymagania krytyczne dla systemu. Muszą zostać zrealizowane w bieżącej wersji, aby system mógł zostać uznany za ukończony i spełniał podstawowe cele biznesowe.

**S – *Should have*** wymagania bardzo ważne. Powinny zostać zrealizowane, jeśli pozwolą na to dostępne zasoby (czas, zespół), jednak w sytuacji konieczności ograniczenia zakresu mogą zostać przesunięte do kolejnego wydania.

**C – *Could have*** wymagania opcjonalne, „mile widziane”. Zwiększają wygodę, kompletność lub atrakcyjność systemu, ale ich brak nie uniemożliwia osiągnięcia głównych celów projektu.

**W – *Won't have this time*** wymagania świadomie odłożone. Zostały zidentyfikowane, jednak nie będą realizowane w obecnym zakresie projektu (bieżącej wersji systemu); mogą stanowić bazę dla przyszłego rozwoju rozwiązania.

W dalszej części rozdziału każdy opis wymagania zawiera przypisany priorytet MoSCoW zgodnie z powyższą klasyfikacją.

Ponadto wymagania mogą mieć jeden z dwóch statusów realizacji:

**Zrealizowano** – zadanie zostało zrealizowane.

**Anulowano** – zadanie zostało anulowane.

## 4.1 Przypadki użycia

### 4.1.1 Aktorzy

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

**Użytkownik (nie premium)** Zarejestrowany użytkownik: zarządza kontem i ulubionymi spotami, dodaje treści/komentarze, korzysta z czatu.

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

**Moderator** Moderacja treści: posty na forum, komentarze spotów.

**Deweloper** Rozwija i utrzymuje system.

#### Aktorzy będący zewnętrznymi usługami

Poniżej wymieniono aktorów, których opisy zamieszczono w rozdziale poświęconym integracji z zewnętrznymi usługami [3.4.3](#).

- Usługa mailowa (Mailtrap)
- Dostawca API do map (OpenFreeMap)
- Nawigacja (Google Maps)
- Dostawca API pogodowego (Open-Meteo)
- Dostawca API GIFów (Tenor)
- Dostawca API do określania strefy czasowej spota (“Where the ISS at?”)

- Dostawca API do geolokalizacji (LocationIQ)
- Azure Blob Storage
- Dostawca OAuth (Google)
- Dostawca OAuth (GitHub)

#### 4.1.2 Diagram przypadków użycia

#### 4.1.3 Scenariusz przypadków użycia

### 4.2 Wymagania

W niniejszym rozdziale przedstawiono wymagania stawiane projektowanemu systemowi. Celem rozdziału jest zebranie w jednym miejscu oczekiwań interesariuszy oraz usystematyzowanie ich w postaci jasno zdefiniowanych kategorii wymagań.

W pracy wyróżniono następujące typy wymagań:

- **Wymagania ogólne** – opisują system na wysokim poziomie, z perspektywy celu biznesowego i użytkownika, bez wchodzenia w szczegóły techniczne. Określają, jakie główne problemy ma rozwiązywać system i jakie korzyści ma dostarczać interesariuszom.
- **Wymagania dziedzinowe** – wynikają ze specyfiki obszaru, w którym wykorzystywany jest system (domena problemu). Odzwierciedlają reguły biznesowe, ograniczenia prawne oraz ustaloną praktykę w danej dziedzinie i muszą być spełnione niezależnie od przyjętych rozwiązań technicznych.
- **Wymagania funkcjonalne** – opisują konkretne funkcje systemu widziane z perspektywy użytkownika lub innego aktora. Definiują, jakie operacje system ma umożliwiać, jakie dane przetwarzać oraz jak reagować na określone zdarzenia i scenariusze użycia.
- **Wymagania pozafunkcjonalne** – określają, *jak* system ma realizować swoje funkcje, a nie *co* ma robić. Obejmują m.in. wymagania dotyczące



wydajności, bezpieczeństwa, niezawodności, użyteczności, skalowalności oraz utrzymywalności rozwiązania.

- **Wymagania dotyczące interfejsu z otoczeniem** – opisują sposób komunikacji systemu z innymi systemami, urządzeniami lub usługami zewnętrznymi. Określają format wymienianych danych, wykorzystywane protokoły, kierunek i częstotliwość wymiany informacji oraz wymagania dotyczące integracji.
- **Wymagania dotyczące środowiska docelowego** – definiują warunki techniczne, w jakich system ma być uruchamiany i eksploatowany. Dotyczą m.in. platformy sprzętowej i programowej, systemów operacyjnych, zasobów sieciowych oraz innych elementów infrastruktury niezbędnych do poprawnego działania aplikacji. (TODO do usunięcia?)

W dalszej części rozdziału wymagania zostały logicznie pogrupowane według obszarów funkcjonalnych systemu: przedstawiono wymagania ogólne, a następnie wymagania dotyczące czatu, forum, mapy oraz panelu użytkownika.

## 4.2.1 Wymagania ogólne

### 4.2.1.1 Wymagania ogólne dla czatu

KARTA WYMAGANIA OGÓLNEGO DLA CZATU			
<b>Identyfikator:</b>	WOCZAT-01	<b>Priorytet:</b>	S
<b>Nazwa:</b>	Wysyłanie wiadomości na czacie		
<b>Opis:</b>	System umożliwia użytkownikowi wysyłanie wiadomości w ramach wybranego czatu, tak aby uczestnicy mogli przekazywać sobie wiedzę na temat dronów lub umawiać się na wspólne spotkania w danym spocie.		
<b>Udziałowiec:</b>	UO3		
<b>Wymagania powiązane:</b>	— brak		

**Tabela 4.1:** Karta wymagania ogólnego dla czatu: Wysyłanie wiadomości na czacie

KARTA WYMAGANIA OGÓLNEGO DLA CZATU			
<b>Identyfikator:</b>	WOCZAT-02	<b>Priorytet:</b>	S
<b>Nazwa:</b>	Edycja czatu		
<b>Opis:</b>	System umożliwia użytkownikowi z odpowiednimi uprawnieniami wprowadzanie podstawowych zmian w konfiguracji czatu, takich jak nazwa czatu czy skład uczestników.		
<b>Udziałowiec:</b>	UO3		
<b>Wymagania powiązane:</b>	— brak		

**Tabela 4.2:** Karta wymagania ogólnego dla czatu: Edycja czatu

KARTA WYMAGANIA OGÓLNEGO DLA CZATU			
<b>Identyfikator:</b>	WOCZAT-03	<b>Priorytet:</b>	S
<b>Nazwa:</b>	Przeglądanie historii czatu		
<b>Opis:</b>	System udostępnia użytkownikowi możliwość przeglądania wcześniejszych wiadomości na czacie, tak aby mógł wracać do poprzednich rozmów i ustaleń.		
<b>Udziałowiec:</b>	UO3		
<b>Wymagania powiązane:</b>	— brak		

**Tabela 4.3:** Karta wymagania ogólnego dla czatu: Przeglądanie historii czatu

KARTA WYMAGANIA OGÓLNEGO DLA CZATU			
<b>Identyfikator:</b>	WOCZAT-04	<b>Priorytet:</b>	S
<b>Nazwa:</b>	Tworzenie czatu		

KARTA WYMAGANIA OGÓLNEGO DLA CZATU (cd.)	
<b>Opis:</b>	System umożliwia użytkownikowi inicjowanie nowych rozmów poprzez tworzenie czatów prywatnych (1:1) lub grupowych z wybranymi uczestnikami.
<b>Udziałowiec:</b>	UO3
<b>Wymagania powiązane:</b>	— brak

**Tabela 4.4:** Karta wymagania ogólnego dla czatu: Tworzenie czatu

## 4.2.2 Wymagania funkcjonalne

Niniejszy rozdział zawiera wymagania funkcjonalne postawione systemowi. Został on podzielony tematycznie.

### 4.2.2.1 Funkcjonalności dla mapy

#### 4.2.2.2 Wymagania funkcjonalne dla czatu

KARTA WYMAGANIA FUNKCJONALNEGO DLA CZATU			
<b>Identyfikator:</b>	WFCZAT-01	<b>Priorytet:</b>	M
<b>Nazwa:</b>	Wysyłanie GIF-ów		
<b>Opis:</b>	System umożliwia wysyłanie w wiadomościach animowanych obrazów GIF w ramach wybranego czatu (1:1 lub grupowego).		

KARTA WYMAGANIA FUNKCJONALNEGO DLA CZATU (cd.)	
Kryteria akceptacji:	<ul style="list-style-type: none"> <li>• Użytkownik może wybrać animowany obraz GIF z dysku lub z wbudowanego pickera i dołączyć go do wiadomości.</li> <li>• Po wysłaniu GIF wyświetla się poprawnie w treści czatu u wszystkich uczestników.</li> <li>• Błędne lub zbyt duże pliki GIF są odrzucane z czytelnym komunikatem o błędzie.</li> </ul>
Dane wejściowe:	Identyfikator czatu, identyfikator nadawcy, wybrany GIF (plik lub identyfikator z usługi zewnętrznej).
Warunki początkowe:	Użytkownik jest zalogowany i jest członkiem danego czatu; połączenie z serwerem jest aktywne.
Warunki końcowe:	Wiadomość z GIF-em jest zapisana w historii czatu i widoczna dla wszystkich uprawnionych uczestników.
Sytuacje wyjątkowe:	Brak połączenia z serwerem, przekroczony limit rozmiaru pliku, nieobsługiwany format, brak uprawnień do czatu.
Szczegóły implementacji:	Rozszerzenie mutacji GraphQL <code>createChatMessage</code> o typ wiadomości <code>GIF</code> ; przechowywanie pliku w usłudze składowania plików (np. Azure Blob Storage) lub przechowywanie identyfikatora GIF-a z usługi zewnętrznej.
Udziałowiec:	Użytkownik zalogowany.
Realizator:	Adam Langmesser
Status:	Zrealizowano

Tabela 4.1: Wymaganie funkcjonalne dla czatu: Wysyłanie GIF-ów

KARTA WYMAGANIA FUNKCJONALNEGO DLA CZATU			
Identyfikator:	WFCZAT-02	Priorytet:	M
Nazwa:	Wysyłanie plików		

KARTA WYMAGANIA FUNKCJONALNEGO DLA CZATU (cd.)	
<b>Opis:</b>	System umożliwia dołączanie i wysyłanie plików (np. dokumentów, zdjęć, archiwów) jako załączników do wiadomości na czacie.
<b>Kryteria akceptacji:</b>	<ul style="list-style-type: none"> <li>• Użytkownik może wybrać jeden lub wiele plików z dysku i dołączyć je do wiadomości.</li> <li>• Odbiorca widzi w czacie element reprezentujący plik (nazwa, rozmiar, ikona typu).</li> <li>• Kliknięcie w załącznik umożliwia pobranie lub otwarcie pliku.</li> </ul>
<b>Dane wejściowe:</b>	Identyfikator czatu, identyfikator nadawcy, co najmniej jeden plik do wysłania.
<b>Warunki początkowe:</b>	Użytkownik jest zalogowany, należy do danego czatu; plik nie przekracza ustalonego limitu rozmiaru i jest dozwolonego typu.
<b>Warunki końcowe:</b>	Wiadomość z załącznikiem jest zapisana w systemie, a plik jest dostępny do pobrania dla uczestników czatu.
<b>Sytuacje wyjątkowe:</b>	Przekroczony limit rozmiaru, nieobsługiwany typ pliku, błąd przesyłania, brak miejsca w magazynie plików.
<b>Szczegóły implementacji:</b>	Przesyłanie plików przez dedykowany endpoint uploadu lub część mutacji <code>createChatMessage</code> ; składowanie plików w zewnętrznej usłudze (np. Azure Blob Storage) i przechowywanie w bazie jedynie metadanych.
<b>Udziałowiec:</b>	Użytkownik zalogowany.
<b>Realizator:</b>	Adam Langmesser
<b>Status:</b>	Zrealizowano

**Tabela 4.2:** Wymaganie funkcjonalne dla czatu: Wysyłanie plików

KARTA WYMAGANIA FUNKCJONALNEGO DLA CZATU			
<b>Identyfikator:</b>	WFCZAT-03	<b>Priorytet:</b>	M
<b>Nazwa:</b>	Wysyłanie wiadomości prywatnych		
<b>Opis:</b>	System umożliwia prowadzenie prywatnych rozmów 1:1 pomiędzy dwoma użytkownikami platformy Merkury.		
<b>Kryteria akceptacji:</b>	<ul style="list-style-type: none"> <li>• Użytkownik może rozpocząć nowy czat 1:1 z innym użytkownikiem lub kontynuować istniejący.</li> <li>• Wiadomości z prywatnego czatu są widoczne wyłącznie dla tych dwóch użytkowników.</li> <li>• Nowe wiadomości pojawiają się w czasie zbliżonym do rzeczywistego bez konieczności przeładowania strony.</li> </ul>		
<b>Dane wejściowe:</b>	Identyfikator nadawcy, identyfikator odbiorcy, treść wiadomości oraz ewentualne załączniki.		
<b>Warunki początkowe:</b>	Obaj użytkownicy posiadają aktywne konta i nie zablokowali się nawzajem.		
<b>Warunki końcowe:</b>	Wiadomości prywatne są zapisane w historii czatu 1:1 i dostępne po ponownym otwarciu rozmowy.		
<b>Sytuacje wyjątkowe:</b>	Brak uprawnień (np. zablokowany użytkownik), brak połączenia z serwerem, błąd walidacji treści.		
<b>Szczegóły implementacji:</b>	Wydzielenie typu czatu <i>PRIVATE</i> w modelu domenowym; mutacja <code>createChat</code> dla utworzenia czatu 1:1 oraz <code>createChatMessage</code> dla wysyłania kolejnych wiadomości.		
<b>Udziałowiec:</b>	Użytkownik zalogowany.		
<b>Realizator:</b>	Adam Langmesser		
<b>Status:</b>	Zrealizowano		

**Tabela 4.3:** Wymaganie funkcjonalne dla czatu: Wysyłanie wiadomości prywatnych

KARTA WYMAGANIA FUNKCJONALNEGO DLA CZATU			
<b>Identyfikator:</b>	WFCZAT-04	<b>Priorytet:</b>	M
<b>Nazwa:</b>	Wysyłanie wiadomości do wielu osób jednocześnie		
<b>Opis:</b>	System umożliwia wysyłanie jednej wiadomości do wielu użytkowników w ramach czatu grupowego.		
<b>Kryteria akceptacji:</b>	<ul style="list-style-type: none"> <li>• Wiadomość wysłana w czacie grupowym jest dostarczana wszystkim jego członkom.</li> <li>• Interfejs wyraźnie wskazuje, że rozmowa to czat grupowy (np. nazwa, avatar, liczba uczestników).</li> <li>• Nowi uczestnicy dołączeni do czatu widzą historię rozmowy zgodnie z przyjętą polityką prywatności.</li> </ul>		
<b>Dane wejściowe:</b>	Identyfikator czatu grupowego, identyfikator nadawcy, treść wiadomości oraz ewentualne załączniki.		
<b>Warunki początkowe:</b>	Czat typu grupowego istnieje, a użytkownik wysyłający wiadomość jest jego członkiem.		
<b>Warunki końcowe:</b>	Wiadomość jest zapisana w historii czatu i widoczna dla wszystkich aktualnych uczestników.		
<b>Sytuacje wyjątkowe:</b>	Brak członkostwa nadawcy w czacie, błąd autoryzacji, przekroczenie limitu użytkowników w czacie.		
<b>Szczegóły implementacji:</b>	Typ czatu <i>GROUP</i> w modelu domenowym; wiadomości powiązane z jednym czatem i wieloma użytkownikami; dystrybucja powiadomień i aktualizacji po WebSockets.		
<b>Udziałowiec:</b>	Użytkownik zalogowany.		
<b>Realizator:</b>	Adam Langmesser		
<b>Status:</b>	Zrealizowano		

**Tabela 4.4:** Wymaganie funkcjonalne dla czatu: Wysyłanie wiadomości do wielu osób jednocześnie

KARTA WYMAGANIA FUNKCJONALNEGO DLA CZATU			
<b>Identyfikator:</b>	WFCZAT-05	<b>Priorytet:</b>	M
<b>Nazwa:</b>	Rozpoczynanie nowego czatu		
<b>Opis:</b>	System umożliwia użytkownikowi utworzenie nowego czatu prywatnego lub grupowego oraz dodanie do niego wskazanych uczestników.		
<b>Kryteria akceptacji:</b>	<ul style="list-style-type: none"> <li>• Użytkownik może zainicjować nowy czat z widoku listy czatów lub profilu innego użytkownika.</li> <li>• Po utworzeniu czatu pojawia się na liście czatów wszystkich jego członków.</li> <li>• Uczestnicy mogą natychmiast rozpocząć wymianę wiadomości.</li> </ul>		
<b>Dane wejściowe:</b>	Typ czatu (prywatny/grupowy), nazwa czatu (dla grup), lista identyfikatorów uczestników, identyfikator tworzącego użytkownika.		
<b>Warunki początkowe:</b>	Użytkownik jest zalogowany i ma uprawnienia do zakładania nowych czatów.		
<b>Warunki końcowe:</b>	Nowy czat jest zapisany w bazie danych, powiązany z uczestnikami i widoczny w ich listach czatów.		
<b>Sytuacje wyjątkowe:</b>	Próba utworzenia czatu z nieistniejącym użytkownikiem, przekroczenie maksymalnej liczby uczestników, błąd zapisu w bazie.		
<b>Szczegóły implementacji:</b>	Mutacja GraphQL <code>createChat</code> ; walidacja listy uczestników oraz ewentualnej unikalności czatów 1:1.		
<b>Udziałowiec:</b>	Użytkownik zalogowany.		
<b>Realizator:</b>	Adam Langmesser		
<b>Status:</b>	Zrealizowano		

**Tabela 4.5:** Wymaganie funkcjonalne dla czatu: Rozpoczynanie nowego czatu



KARTA WYMAGANIA FUNKCJONALNEGO DLA CZATU			
<b>Identyfikator:</b>	WFCZAT-06	<b>Priorytet:</b>	M
<b>Nazwa:</b>	Wysyłanie emotikonów		
<b>Opis:</b>	System umożliwia wysyłanie emotikonów (emoji) w treści wiadomości na czacie.		
<b>Kryteria akceptacji:</b>	<ul style="list-style-type: none"> <li>• Użytkownik może wstawiać emotikony z wbudowanego pickera emoji.</li> <li>• Emotikony są poprawnie renderowane na różnych urządzeniach (desktop, mobile).</li> <li>• Kopiowanie lub edycja wiadomości zachowuje wstawione emotikony.</li> </ul>		
<b>Dane wejściowe:</b>	Identyfikator czatu, identyfikator nadawcy, treść wiadomości zawierająca emoji.		
<b>Warunki początkowe:</b>	Użytkownik jest zalogowany i ma dostęp do czatu.		
<b>Warunki końcowe:</b>	Wiadomość z emotikonami jest zapisana w historii czatu i wyświetlana u wszystkich uczestników.		
<b>Sytuacje wyjątkowe:</b>	Problemy z kodowaniem znaków, brak wsparcia dla części emoji w danej przeglądarce lub systemie.		
<b>Szczegóły implementacji:</b>	Wykorzystanie znaków Unicode dla emoji; podpięcie komponentu pickera (np. <code>emoji-mart</code> ) po stronie frontendu.		
<b>Udziałowiec:</b>	Użytkownik zalogowany.		
<b>Realizator:</b>	Adam Langmesser		
<b>Status:</b>	Zrealizowano		

Tabela 4.6: Wymaganie funkcjonalne dla czatu: Wysyłanie emotikonów

KARTA WYMAGANIA FUNKCJONALNEGO DLA CZATU			
<b>Identyfikator:</b>	WFCZAT-07	<b>Priorytet:</b>	M
<b>Nazwa:</b>	Dostępność czatu po utworzeniu		

KARTA WYMAGANIA FUNKCJONALNEGO DLA CZATU (cd.)	
<b>Opis:</b>	Czat po utworzeniu pozostaje dostępny dla jego członków; użytkownik może później odnaleźć czat i wrócić do wcześniejszej konwersacji.
<b>Kryteria akceptacji:</b>	<ul style="list-style-type: none"> <li>• Utworzone czaty pojawiają się na liście czatów użytkownika.</li> <li>• Po ponownym zalogowaniu użytkownik może otworzyć istniejący czat i zobaczyć jego historię.</li> <li>• Usunięcie użytkownika z czatu powoduje usunięcie go z jego listy czatów.</li> </ul>
<b>Dane wejściowe:</b>	Kontekst zalogowanego użytkownika oraz parametry filtrów listy czatów.
<b>Warunki początkowe:</b>	Czat został wcześniej utworzony, a użytkownik jest jego członkiem.
<b>Warunki końcowe:</b>	Lista czatów użytkownika prezentuje wszystkie czaty, do których ma on dostęp.
<b>Sytuacje wyjątkowe:</b>	Błąd autoryzacji (sesja wygasła), błąd bazy danych podczas pobierania listy czatów.
<b>Szczegóły implementacji:</b>	Zapytanie GraphQL <code>chatsByCurrentUser</code> wykorzystywane w panelu bocznym listy czatów; cache po stronie klienta.
<b>Udziałowiec:</b>	Użytkownik zalogowany.
<b>Realizator:</b>	Adam Langmesser
<b>Status:</b>	Zrealizowano

**Tabela 4.7:** Wymaganie funkcjonalne dla czatu: Dostępność czatu po utworzeniu

KARTA WYMAGANIA FUNKCJONALNEGO DLA CZATU			
<b>Identyfikator:</b>	WFCZAT-08	<b>Priorytet:</b>	M
<b>Nazwa:</b>	Edytowanie nazwy czatu grupowego		
<b>Opis:</b>	System umożliwia zmianę nazwy istniejącego czatu grupowego przez jego właściciela lub uprawnionego administratora.		
<b>Kryteria akceptacji:</b>	<ul style="list-style-type: none"> <li>• Uprawniony użytkownik może edytować nazwę z poziomu ustawień czatu.</li> <li>• Nowa nazwa jest natychmiast widoczna na liście czatów u wszystkich uczestników.</li> <li>• Historia wiadomości pozostaje niezmieniona po zmianie nazwy czatu.</li> </ul>		
<b>Dane wejściowe:</b>	Identyfikator czatu grupowego, nowa nazwa czatu, identyfikator użytkownika wykonującego operację.		
<b>Warunki początkowe:</b>	Użytkownik posiada uprawnienia właściciela/administratora w danym czacie grupowym.		
<b>Warunki końcowe:</b>	Nazwa czatu jest zaktualizowana w bazie danych i w interfejsie użytkownika.		
<b>Sytuacje wyjątkowe:</b>	Brak uprawnień do edycji, zbyt długa lub pusta nazwa, błąd zapisu w bazie danych.		
<b>Szczegóły implementacji:</b>	Mutacja GraphQL <code>updateChatMetadata</code> z polem <code>name</code> ; walidacja uprawnień po stronie backendu.		
<b>Udziałowiec:</b>	Właściciel i uczestnicy czatu grupowego.		
<b>Realizator:</b>	Adam Langmesser		
<b>Status:</b>	Zrealizowano		

**Tabela 4.8:** Wymaganie funkcjonalne dla czatu: Edytowanie nazwy czatu grupowego

KARTA WYMAGANIA FUNKCJONALNEGO DLA CZATU			
<b>Identyfikator:</b>	WFCZAT-09	<b>Priorytet:</b>	M
<b>Nazwa:</b>	Edycja zdjęcia czatu grupowego		
<b>Opis:</b>	System umożliwia zmianę obrazu/avatara reprezentującego czat grupowy.		
<b>Kryteria akceptacji:</b>	<ul style="list-style-type: none"> <li>• Właściciel lub administrator może wgrać nowe zdjęcie czatu grupowego.</li> <li>• Zmienione zdjęcie jest widoczne na liście czatów i w nagłówku rozmowy.</li> <li>• Niepoprawne formaty lub zbyt duże pliki są odrzucane z informacją o błędzie.</li> </ul>		
<b>Dane wejściowe:</b>	Identyfikator czatu grupowego, nowy plik graficzny, identyfikator użytkownika wykonującego operację.		
<b>Warunki początkowe:</b>	Użytkownik posiada uprawnienia do edycji ustawień czatu grupowego.		
<b>Warunki końcowe:</b>	Nowe zdjęcie czatu jest zapisane w systemie i używane w interfejsie.		
<b>Sytuacje wyjątkowe:</b>	Nieobsługiwany format obrazu, przekroczony limit rozmiaru, błąd przesyłania lub zapisu w magazynie plików.		
<b>Szczegóły implementacji:</b>	Pole avatarUrl w encji czatu; przesyłanie pliku do usługi składowania plików; aktualizacja adresu URL w bazie danych.		
<b>Udziałowiec:</b>	Właściciel i uczestnicy czatu grupowego.		
<b>Realizator:</b>	Adam Langmesser		
<b>Status:</b>	Zrealizowano		

**Tabela 4.9:** Wymaganie funkcjonalne dla czatu: Edycja zdjęcia czatu grupowego

KARTA WYMAGANIA FUNKCJONALNEGO DLA CZATU			
<b>Identyfikator:</b>	WFCZAT-10	<b>Priorytet:</b>	M
<b>Nazwa:</b>	Edycja wysłanej wiadomości		
<b>Opis:</b>	System umożliwia użytkownikowi edycję treści wcześniej wysłanej wiadomości na czacie (np. poprawa literówki).		
<b>Kryteria akceptacji:</b>	<ul style="list-style-type: none"> <li>• Użytkownik może edytować swoje wiadomości przez ograniczony czas od momentu wysłania (konfigurowalny limit).</li> <li>• Zmieniona wiadomość jest oznaczona etykietą „(edytowano)”.</li> <li>• Odbiorcy widzą zaktualizowaną treść bez duplikowania wiadomości.</li> </ul>		
<b>Dane wejściowe:</b>	Identyfikator wiadomości, nowa treść wiadomości, identyfikator użytkownika wykonującego edycję.		
<b>Warunki początkowe:</b>	Użytkownik jest autorem wiadomości i mieści się w dozwolonym oknie czasowym edycji.		
<b>Warunki końcowe:</b>	Treść wiadomości w bazie danych zostaje zaktualizowana, a widok czatu odświeżony u wszystkich uczestników.		
<b>Sytuacje wyjątkowe:</b>	Próba edycji cudzej wiadomości, przekroczony limit czasu, błąd walidacji treści, brak połączenia z serwerem.		
<b>Szczegóły implementacji:</b>	Mutacja GraphQL <code>updateChatMessage</code> ; przechowywanie znacznika <code>editedAt</code> oraz dystrybucja aktualizacji poprzez WebSocket.		
<b>Udziałowiec:</b>	Użytkownik zalogowany oraz odbiorcy wiadomości.		
<b>Realizator:</b>	Adam Langmesser		
<b>Status:</b>	Zrealizowano		

**Tabela 4.10:** Wymaganie funkcjonalne dla czatu: Edycja wysłanej wiadomości

KARTA WYMAGANIA FUNKCJONALNEGO DLA CZATU			
<b>Identyfikator:</b>	WFCZAT-11	<b>Priorytet:</b>	M
<b>Nazwa:</b>	Usunięcie wysłanej wiadomości		
<b>Opis:</b>	System umożliwia usunięcie wysłanej wiadomości z czatu (dla siebie lub dla wszystkich uczestników, zgodnie z konfiguracją).		
<b>Kryteria akceptacji:</b>	<ul style="list-style-type: none"> <li>• Autor wiadomości może ją usunąć w dozwolonym przedziale czasowym.</li> <li>• W miejscu usuniętej wiadomości może pojawić się informacja typu „Wiadomość usunięta”.</li> <li>• Usunięcie jednej wiadomości nie narusza integralności pozostałej historii czatu.</li> </ul>		
<b>Dane wejściowe:</b>	Identyfikator wiadomości, tryb usunięcia (dla siebie / dla wszystkich), identyfikator użytkownika.		
<b>Warunki początkowe:</b>	Użytkownik jest autorem wiadomości lub posiada uprawnienia moderacyjne w czacie.		
<b>Warunki końcowe:</b>	Wiadomość jest oznaczona jako usunięta i nie jest prezentowana w pierwotnej treści uczestnikom czatu.		
<b>Sytuacje wyjątkowe:</b>	Próba usunięcia cudzej wiadomości bez uprawnień, przekroczony limit czasu, błąd zapisu w bazie danych.		
<b>Szczegóły implementacji:</b>	Mutacja <code>deleteChatMessage</code> lub aktualizacja pola <code>deletedAt</code> ; odświeżenie widoku w kliencie przez WebSocket.		
<b>Udziałowiec:</b>	Użytkownik zalogowany oraz moderatorzy czatu.		
<b>Realizator:</b>	Adam Langmesser		
<b>Status:</b>	Zrealizowano		

**Tabela 4.11:** Wymaganie funkcjonalne dla czatu: Usunięcie wysłanej wiadomości

KARTA WYMAGANIA FUNKCJONALNEGO DLA CZATU			
<b>Identyfikator:</b>	WFCZAT-12	<b>Priorytet:</b>	M
<b>Nazwa:</b>	Dodawanie użytkowników do istniejącego czatu		
<b>Opis:</b>	System umożliwia dodawanie nowych użytkowników do już istniejącego czatu grupowego.		
<b>Kryteria akceptacji:</b>	<ul style="list-style-type: none"> <li>• Właściciel lub uprawniony użytkownik może wyszukać i dodać nowe osoby do czatu.</li> <li>• Nowo dodani użytkownicy pojawiają się na liście uczestników i mogą od razu brać udział w rozmowie.</li> <li>• Dodanie uczestnika jest widoczne dla pozostałych w formie komunikatu systemowego.</li> </ul>		
<b>Dane wejściowe:</b>	Identyfikator czatu grupowego, lista identyfikatorów użytkowników do dodania, identyfikator użytkownika wykonującego operację.		
<b>Warunki początkowe:</b>	Czat typu grupowego istnieje; użytkownik wykonujący operację ma odpowiednie uprawnienia.		
<b>Warunki końcowe:</b>	Lista uczestników czatu jest zaktualizowana; nowe osoby są powiązane z czatem w bazie danych.		
<b>Sytuacje wyjątkowe:</b>	Dodanie użytkownika, który nie ma konta, został zablokowany lub jest już członkiem czatu; przekroczenie maksymalnej liczby uczestników.		
<b>Szczegóły implementacji:</b>	Mutacja GraphQL <code>updateChatParticipants</code> z listą identyfikatorów uczestników; walidacja po stronie backendu.		
<b>Udziałowiec:</b>	Właściciel czatu grupowego oraz jego uczestnicy.		
<b>Realizator:</b>	Adam Langmesser		
<b>Status:</b>	Zrealizowano		

**Tabela 4.12:** Wymaganie funkcjonalne dla czatu: Dodawanie użytkowników do istniejącego czatu

KARTA WYMAGANIA FUNKCJONALNEGO DLA CZATU			
<b>Identyfikator:</b>	WFCZAT-13	<b>Priorytet:</b>	M
<b>Nazwa:</b>	Wyświetlanie starszych wiadomości		
<b>Opis:</b>	System powinien domyślnie wyświetlać co najmniej ostatnie 20 wiadomości w czacie, a starsze wiadomości dociągać na bieżąco podczas przewijania historii przez użytkownika.		
<b>Kryteria akceptacji:</b>	<ul style="list-style-type: none"> <li>Po wejściu na czat użytkownik widzi minimum 20 ostatnich wiadomości.</li> <li>Przewijanie historii w górę automatycznie pobiera starsze wiadomości (mechanizm <i>infinite scroll</i>).</li> <li>Dociąganie wiadomości nie powoduje zauważalnych opóźnień interfejsu.</li> </ul>		
<b>Dane wejściowe:</b>	Identyfikator czatu, parametry paginacji (np. kursor, znacznik czasu).		
<b>Warunki początkowe:</b>	Użytkownik jest zalogowany i posiada dostęp do czatu; w bazie istnieje historia rozmowy.		
<b>Warunki końcowe:</b>	Użytkownik ma możliwość przeglądania pełnej historii czatu w zakresie swoich uprawnień.		
<b>Sytuacje wyjątkowe:</b>	Brak połączenia z serwerem, błąd paginacji, przekroczenie limitu zapytań do API.		
<b>Szczegóły implementacji:</b>	Zapytania GraphQL z paginacją kursorową; komponent <i>infinite scroll</i> oparty na TanStack Query; cache i stronicowanie po stronie klienta.		
<b>Udziałowiec:</b>	Użytkownik zalogowany.		
<b>Realizator:</b>	Adam Langmesser		



KARTA WYMAGANIA FUNKCJONALNEGO DLA CZATU (cd.)	
Status:	Zrealizowano

**Tabela 4.13:** Wymaganie funkcjonalne dla czatu: Wyświetlanie starszych wiadomości

#### 4.2.2.3 Funkcjonalności dla forum

#### 4.2.2.4 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 <code>@tanstack/react-query</code> i <code>axios</code> z <code>withCredentials</code> . Prezentacja w widoku profilu.		
Udziałowiec:	Zespół projektowy 2.1; promotor 2.2; droniarze 2.3.		
Wymagania powiązane:			

Tabela 4.18: 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 4.19: 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 4.20: 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 4.21: 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 4.22: 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 4.23: 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 4.24: 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 4.25: 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 4.26: 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 4.27: 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 4.28: 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 <b>axios</b> ; 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 4.29: 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 4.30: Dodawanie do znajomych

#### 4.2.2.5 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 <code>axios</code> z <code>withCredentials</code> . SSO: integracja z Google i GitHub (OAuth 2.0) z przekierowaniem i ustawieniem sesji po stronie backendu ( <code>httpOnly</code> 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 4.31: Logowanie i rejestracja

#### 4.2.2.6 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 <code>@tanstack/react-query</code> i <code>axios</code> (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 4.32: 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 4.33: Strona główna — zaawansowane filtry

#### 4.2.2.7 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>jasny</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 <a href="#">2.1</a> , promotor <a href="#">2.2</a> , droniarze <a href="#">2.3</a> .		
Wymagania powiązane:			

Tabela 4.34: 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 4.35: Zapamiętanie 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 4.36: Szybki przełącznik motywu w interfejsie

### 4.2.3 Wymagania pozafunkcjonalne

Niniejszy rozdział zawiera wymagania pozafunkcjonalne postawione systemowi. Został on podzielony tematycznie.

#### 4.2.3.1 Wymagania pozafunkcjonalne dla czatu

KARTA WYMAGANIA POZAFUNKCJONALNEGO DLA CZATU			
Identyfikator:	WPCZAT-01	Priorytet:	M
Nazwa:	Ograniczenie widoczności czatów do członków		
Typ:	Bezpieczeństwo		



KARTA WYMAGANIA POZAFUNKCJONALNEGO DLA CZATU (cd.)	
<b>Opis:</b>	System zapewnia, że użytkownik widzi wyłącznie listę czatów oraz wiadomości z czatów, których jest członkiem. Informacje o innych czatach nie są prezentowane w interfejsie ani dostępne poprzez API.
<b>Kryteria akceptacji:</b>	<ul style="list-style-type: none"> <li>• Lista czatów dla zalogowanego użytkownika zawiera wyłącznie czaty, w których jest on uczestnikiem.</li> <li>• Próba otwarcia czatu, którego użytkownik nie jest członkiem, kończy się czytelnym błędem (np. <i>brak uprawnień</i> lub <i>nie znaleziono</i>).</li> <li>• Wiadomości z czatów, do których użytkownik nie ma dostępu, nie są zwracane przez API ani widoczne w logach klienta.</li> </ul>
<b>Dane wejściowe:</b>	Kontekst zalogowanego użytkownika, identyfikator użytkownika, identyfikator czatu (dla widoku szczegółowego).
<b>Warunki początkowe:</b>	Użytkownik jest poprawnie uwierzytelniony; w systemie istnieją zarejestrowane czaty i przypisania użytkownik–czat.
<b>Warunki końcowe:</b>	Użytkownik ma dostęp wyłącznie do własnych czatów i ich wiadomości, zgodnie z zapisanymi członkostwami.
<b>Sytuacje wyjątkowe:</b>	Błędna konfiguracja uprawnień, niespójne dane o członkostwach w bazie, awaria usługi autoryzacji.
<b>Szczegóły implementacji:</b>	Filtrowanie danych po stronie backendu na podstawie identyfikatora użytkownika; wymuszenie sprawdzania członkostwa przy każdym zapytaniu o czat lub wiadomości; testy integracyjne dla scenariuszy <i>brak uprawnień</i> .
<b>Udziałowiec:</b>	Użytkownik zalogowany, administrator systemu.
<b>Realizator:</b>	Adam Langmesser

KARTA WYMAGANIA POZAFUNKCJONALNEGO DLA CZATU (cd.)	
Status:	Zrealizowano

**Tabela 4.1:** Wymaganie pozafunkcjonalne dla czatu: Ograniczenie widoczności czatów do członków

KARTA WYMAGANIA POZAFUNKCJONALNEGO DLA CZATU			
Identyfikator:	WPCZAT-02	Priorytet:	M
Nazwa:	Wymóg zalogowania do korzystania z czatu		
Typ:	Bezpieczeństwo		
Opis:	Dostęp do funkcji czatu (lista czatów, wysyłanie i odbieranie wiadomości, tworzenie czatów) wymaga wcześniejszego zalogowania się do systemu. Użytkownik niezalogowany nie może przeglądać ani modyfikować danych czatu.		
Kryteria akceptacji:	<ul style="list-style-type: none"> <li>• Wejście na widok czatu przez użytkownika niezalogowanego powoduje przekierowanie na stronę logowania lub komunikat o braku uprawnień.</li> <li>• Zapytania do API czatu bez ważnego tokena/autoryzacji są odrzucane.</li> <li>• Po poprawnym zalogowaniu użytkownik uzyskuje pełny dostęp do swoich czatów bez konieczności ponownego logowania w tej sesji.</li> </ul>		
Dane wejściowe:	Dane logowania użytkownika, token sesji lub inny mechanizm uwierzytelniania.		
Warunki początkowe:	Użytkownik posiada aktywne konto w systemie.		
Warunki końcowe:	Tylko użytkownicy zalogowani mogą korzystać z funkcji czatu; użytkownicy niezalogowani widzą jedynie ekran logowania/rejestracji.		

KARTA WYMAGANIA POZAFUNKCJONALNEGO DLA CZATU (cd.)	
<b>Sytuacje wyjątkowe:</b>	Wygaśnięcie sesji, utrata tokena, błąd integracji z modulem logowania.
<b>Szczegóły implementacji:</b>	Middleware/autoryzacja na poziomie backendu wymuszająca uwierzytelnienie dla wszystkich endpointów czatu; przechowywanie danych sesji w bezpieczny sposób (np. token JWT).
<b>Udziałowiec:</b>	Użytkownik zalogowany, administrator systemu.
<b>Realizator:</b>	Adam Langmesser
<b>Status:</b>	Zrealizowano

**Tabela 4.2:** Wymaganie pozafunkcjonalne dla czatu: Wymóg zalogowania do korzystania z czatu

KARTA WYMAGANIA POZAFUNKCJONALNEGO DLA CZATU			
<b>Identyfikator:</b>	WPCZAT-03	<b>Priorytet:</b>	M
<b>Nazwa:</b>	Grupowanie wiadomości według daty wysłania		
<b>Typ:</b>	Użyteczność		
<b>Opis:</b>	Wiadomości na czacie są prezentowane w logicznych grupach odpowiadających datom ich wysłania (np. separatory dzienne), co ułatwia użytkownikom orientację w historii rozmowy.		
<b>Kryteria akceptacji:</b>	<ul style="list-style-type: none"> <li>• W widoku czatu pojawiają się wizualne znaczniki dat (np. „Dzisiaj”, „Wczoraj”, konkretna data).</li> <li>• Wiadomości są zawsze przypisane do poprawnej grupy daty wysłania niezależnie od strefy czasowej klienta.</li> <li>• Zmiana zakresu historii (scrollowanie, przeładowanie) zachowuje poprawne grupowanie dat.</li> </ul>		

KARTA WYMAGANIA POZAFUNKCJONALNEGO DLA CZATU (cd.)	
<b>Dane wejściowe:</b>	Wiadomości danego czatu wraz z ich znacznikami czasu wysłania.
<b>Warunki początkowe:</b>	Wiadomości posiadają poprawnie zapisany czas wysłania w spójnym formacie.
<b>Warunki końcowe:</b>	Użytkownik widzi wiadomości zgrupowane według dat, co poprawia czytelność dłuższych rozmów.
<b>Sytuacje wyjątkowe:</b>	Błędne strefy czasowe lub niepoprawne wartości znaczników czasu w bazie danych.
<b>Szczegóły implementacji:</b>	Normalizacja dat i czasów do strefy referencyjnej (np. UTC) po stronie backendu oraz odpowiednie formatowanie i grupowanie po stronie klienta.
<b>Udziałowiec:</b>	Użytkownik zalogowany.
<b>Realizator:</b>	Adam Langmesser
<b>Status:</b>	Zrealizowano

**Tabela 4.3:** Wymaganie pozafunkcjonalne dla czatu: Grupowanie wiadomości według daty wysłania

KARTA WYMAGANIA POZAFUNKCJONALNEGO DLA CZATU			
<b>Identyfikator:</b>	WPCZAT-04	<b>Priorytet:</b>	M
<b>Nazwa:</b>	Wyraźne oznaczenie nadawcy i czasu wysłania		
<b>Typ:</b>	Użyteczność		
<b>Opis:</b>	Każda wiadomość na czacie jest opatrzona wyraźną informacją, kto jest jej nadawcą oraz kiedy została wysłana (data i czas). Informacje te są łatwo zauważalne i spójne wizualnie.		

KARTA WYMAGANIA POZAFUNKCJONALNEGO DLA CZATU (cd.)	
<b>Kryteria akceptacji:</b>	<ul style="list-style-type: none"> <li>• Przy każdej wiadomości widoczna jest nazwa lub alias nadawcy.</li> <li>• Czas wysłania jest prezentowany w czytelnej formie (np. „12:34”, „wczoraj 21:10”) z uwzględnieniem lokalnej strefy czasowej użytkownika.</li> <li>• Przejście kursorem lub tapnięcie umożliwia wyświetlenie pełnej daty i czasu w bardziej szczegółowym formacie.</li> </ul>
<b>Dane wejściowe:</b>	Dane użytkownika–nadawcy oraz znacznik czasu wysłania każdej wiadomości.
<b>Warunki początkowe:</b>	Dane użytkowników i wiadomości są spójne; każda wiadomość ma powiązanego nadawcę i poprawny czas wysłania.
<b>Warunki końcowe:</b>	Użytkownik może jednoznacznie zidentyfikować nadawcę i czas wysłania każdej wiadomości.
<b>Sytuacje wyjątkowe:</b>	Brak danych o nadawcy lub czasie (stare wiadomości, dane testowe), błędy migracji danych.
<b>Szczegóły implementacji:</b>	Powiązanie wiadomości z tabelą użytkowników po kluczu obcym; formatowanie czasu po stronie klienta; spójny komponent UI dla nagłówka wiadomości.
<b>Udziałowiec:</b>	Użytkownik zalogowany, moderatorzy czatu.
<b>Realizator:</b>	Adam Langmesser
<b>Status:</b>	Zrealizowano

**Tabela 4.4:** Wymaganie pozafunkcjonalne dla czatu: Wyraźne oznaczenie nadawcy i czasu wysłania

KARTA WYMAGANIA POZAFUNKCJONALNEGO DLA CZATU			
<b>Identyfikator:</b>	WPCZAT-05	<b>Priorytet:</b>	M

KARTA WYMAGANIA POZAFUNKCJONALNEGO DLA CZATU (cd.)	
<b>Nazwa:</b>	Czas załadowania starszych wiadomości poniżej 3 sekund
<b>Typ:</b>	Wydajność
<b>Opis:</b>	Podczas przewijania historii czatu załadowanie kolejnej partii starszych wiadomości powinno trwać krócej niż 3 sekundy w typowych warunkach sieciowych.
<b>Kryteria akceptacji:</b>	<ul style="list-style-type: none"> <li>• W co najmniej 95% pomiarów laboratoryjnych i testów akceptacyjnych czas pobrania starszych wiadomości mieści się w przedziale 0–3 s.</li> <li>• Interfejs wyraźnie sygnalizuje trwające ładowanie (np. animacja „ładowanie”), a po zakończeniu przewijanie jest płynne.</li> <li>• Brak zauważalnych „zawieszeń” interfejsu podczas dociągania danych.</li> </ul>
<b>Dane wejściowe:</b>	Identyfikator czatu, parametry paginacji (np. kursor, znacznik czasu), liczba wiadomości do pobrania.
<b>Warunki początkowe:</b>	W bazie istnieje historia rozmowy; serwer i baza danych działają poprawnie.
<b>Warunki końcowe:</b>	Starsze wiadomości są dołączone do historii czatu użytkownika w czasie krótszym niż 3 s w typowych warunkach.
<b>Sytuacje wyjątkowe:</b>	Bardzo słabe łącze użytkownika, awaria sieci, wysoka chwilowa niedostępność bazy danych.
<b>Szczegóły implementacji:</b>	Indeksy po znaczniku czasu w bazie danych, paginacja kursorowa, cache po stronie serwera i klienta; ograniczenie liczby pobieranych rekordów w jednym żądaniu.
<b>Udziałowiec:</b>	Użytkownik zalogowany.
<b>Realizator:</b>	Adam Langmesser
<b>Status:</b>	Zrealizowano

**Tabela 4.5:** Wymaganie pozafunkcjonalne dla czatu: Czas załadowania starszych wiadomości poniżej 3 sekund

KARTA WYMAGANIA POZAFUNKCJONALNEGO DLA CZATU			
<b>Identyfikator:</b>	WPCZAT-06	<b>Priorytet:</b>	M
<b>Nazwa:</b>	Natychmiastowe wysyłanie wiadomości		
<b>Typ:</b>	Wydajność		
<b>Opis:</b>	Po wysłaniu wiadomości przez użytkownika powinna ona pojawić się w widoku czatu w czasie subiektywnie natychmiastowym (rzędu setek milisekund), a pozostali uczestnicy powinni ją zobaczyć w czasie zbliżonym do rzeczywistego.		
<b>Kryteria akceptacji:</b>	<ul style="list-style-type: none"> <li>• W typowych warunkach sieciowych użytkownik widzi swoją nową wiadomość w czacie w czasie poniżej 1 s od wysłania.</li> <li>• Pozostali uczestnicy czatu otrzymują wiadomość bez konieczności ręcznego odświeżania (np. przez WebSocket).</li> <li>• W przypadku chwilowego opóźnienia interfejs sygnalizuje status wysyłania (np. ikona „wysyłanie...”, „niedostarczona”).</li> </ul>		
<b>Dane wejściowe:</b>	Identyfikator czatu, identyfikator nadawcy, treść wiadomości oraz ewentualne załączniki.		
<b>Warunki początkowe:</b>	Użytkownik jest zalogowany, posiada dostęp do czatu, serwer jest dostępny.		
<b>Warunki końcowe:</b>	Wiadomość jest zapisana w bazie danych oraz dostarczona wszystkim uprawnionym uczestnikom czatu.		
<b>Sytuacje wyjątkowe:</b>	Zerwanie połączenia sieciowego, chwilowa niedostępność serwera, przeciążenie systemu.		

KARTA WYMAGANIA POZAFUNKCJONALNEGO DLA CZATU (cd.)	
<b>Szczegóły implementacji:</b>	Wykorzystanie połączeń WebSocket lub innego kanału push; kolejka wiadomości po stronie serwera; asynchroniczne zapisy do bazy danych.
<b>Udziałowiec:</b>	Użytkownik zalogowany.
<b>Realizator:</b>	Adam Langmesser
<b>Status:</b>	Zrealizowano

**Tabela 4.6:** Wymaganie pozafunkcjonalne dla czatu: Natychmiastowe wysyłanie wiadomości

KARTA WYMAGANIA POZAFUNKCJONALNEGO DLA CZATU			
<b>Identyfikator:</b>	WPCZAT-07	<b>Priorytet:</b>	M
<b>Nazwa:</b>	Zachowanie wiadomości przy chwilowej utracie połączenia		
<b>Typ:</b>	Niezawodność		
<b>Opis:</b>	W przypadku krótkotrwałej utraty połączenia sieciowego wiadomości wysłane przez użytkownika nie powinny zostać utracone: zostaną ponownie wysłane po odzyskaniu łączności lub jednoznacznie oznaczone jako niedostarczone.		
<b>Kryteria akceptacji:</b>	<ul style="list-style-type: none"> <li>• Wiadomości wysłane w momencie utraty połączenia są buforowane lokalnie do czasu ponownej próby wysłania.</li> <li>• Po odzyskaniu połączenia następuje automatyczna próba ponownego dostarczenia buforowanych wiadomości.</li> <li>• W przypadku braku możliwości dostarczenia użytkownik otrzymuje czytelną informację (status „nie-dostarczona”) i może spróbować ponownie.</li> </ul>		



KARTA WYMAGANIA POZAFUNKCJONALNEGO DLA CZATU (cd.)	
<b>Dane wejściowe:</b>	Bufor lokalny wiadomości po stronie klienta, status połączenia sieciowego, identyfikator czatu i nadawcy.
<b>Warunki początkowe:</b>	Użytkownik jest zalogowany; przed utratą połączenia czat działał poprawnie.
<b>Warunki końcowe:</b>	Wiadomości wysłane w okresie chwilowej utraty połączenia są ostatecznie dostarczone lub jasno oznaczone jako niedostarczone, bez „cichej” utraty danych.
<b>Sytuacje wyjątkowe:</b>	Długotrwały brak połączenia, ręczne zamknięcie aplikacji przed synchronizacją bufora, usunięcie danych lokalnych.
<b>Szczegóły implementacji:</b>	Bufor wysyłanych wiadomości w pamięci lub local storage po stronie klienta; mechanizm ponawiania wysyłki; idempotentne mutacje po stronie backendu.
<b>Udziałowiec:</b>	Użytkownik zalogowany.
<b>Realizator:</b>	Adam Langmesser
<b>Status:</b>	Zrealizowano

**Tabela 4.7:** Wymaganie pozafunkcjonalne dla czatu: Zachowanie wiadomości przy chwilowej utracie połączenia

KARTA WYMAGANIA POZAFUNKCJONALNEGO DLA CZATU			
<b>Identyfikator:</b>	WPCZAT-08	<b>Priorytet:</b>	M
<b>Nazwa:</b>	Limit wysyłanych wiadomości w jednostce czasu		
<b>Typ:</b>	Niezawodność		
<b>Opis:</b>	System ogranicza maksymalną liczbę wiadomości wysyłanych przez pojedynczego użytkownika w określonej jednostce czasu, aby chronić czat przed spamem oraz przeciążeniem.		

KARTA WYMAGANIA POZAFUNKCJONALNEGO DLA CZATU (cd.)	
<b>Kryteria akceptacji:</b>	<ul style="list-style-type: none"> <li>• Przy zbyt częstym wysyłaniu wiadomości użytkownik otrzymuje informację o osiągnięciu limitu, a kolejne wiadomości są tymczasowo blokowane.</li> <li>• Normalna praca czatu nie jest utrudniona dla typowych wzorców użycia (pisanie „normalnym tempem” nie wyczerpuje limitu).</li> <li>• Limity mogą być konfigurowane (np. inne dla zwykłych użytkowników i administratorów).</li> </ul>
<b>Dane wejściowe:</b>	Identyfikator użytkownika, znaczniki czasu wysłanych wiadomości w ostatnim przedziale czasowym.
<b>Warunki początkowe:</b>	Użytkownik jest zalogowany; moduł czatu rejestruje czas wysłania każdej wiadomości.
<b>Warunki końcowe:</b>	Nadmierne tempo wysyłania wiadomości przez jednego użytkownika jest ograniczane, co zmniejsza ryzyko przeciążenia systemu i spamu.
<b>Sytuacje wyjątkowe:</b>	Błędy konfiguracji limitów, ataki rozproszone z wielu kont, duże wahania ruchu w krótkim czasie.
<b>Szczegóły implementacji:</b>	Mechanizm <i>rate limiting</i> na poziomie backendu (np. licznik w pamięci lub magazynie typu Redis) powiązany z identyfikatorem użytkownika lub adresu IP; zwracanie odpowiedniego kodu błędu przy przekroczeniu limitu.
<b>Udziałowiec:</b>	Użytkownik zalogowany, administrator systemu.
<b>Realizator:</b>	Adam Langmesser
<b>Status:</b>	Zrealizowano

**Tabela 4.8:** Wymaganie pozafunkcjonalne dla czatu: Limit wysyłanych wiadomości w jednostce czasu

# Rozdział 5

## Projekt

### 5.1 Wzorce projektowe

### 5.2 Architektura systemu

#### 5.2.1 Diagram architektury

#### 5.2.2 Komponenty systemu

### 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

#### 7.1.1 Struktura projektu

#### 7.1.2 Integracja z bazą danych

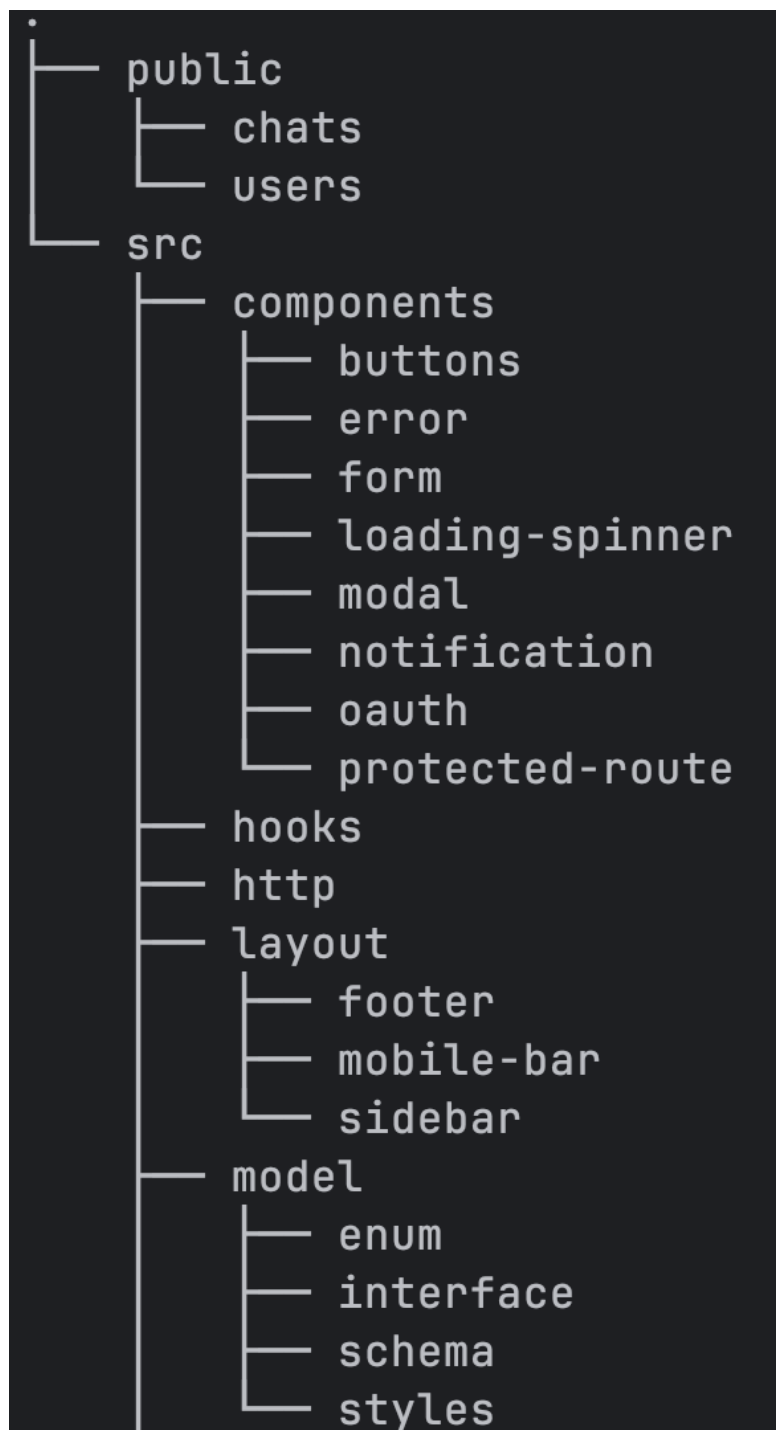
#### 7.1.3 Obsługa uwierzytelnienia

#### 7.1.4 Konteneryzacja

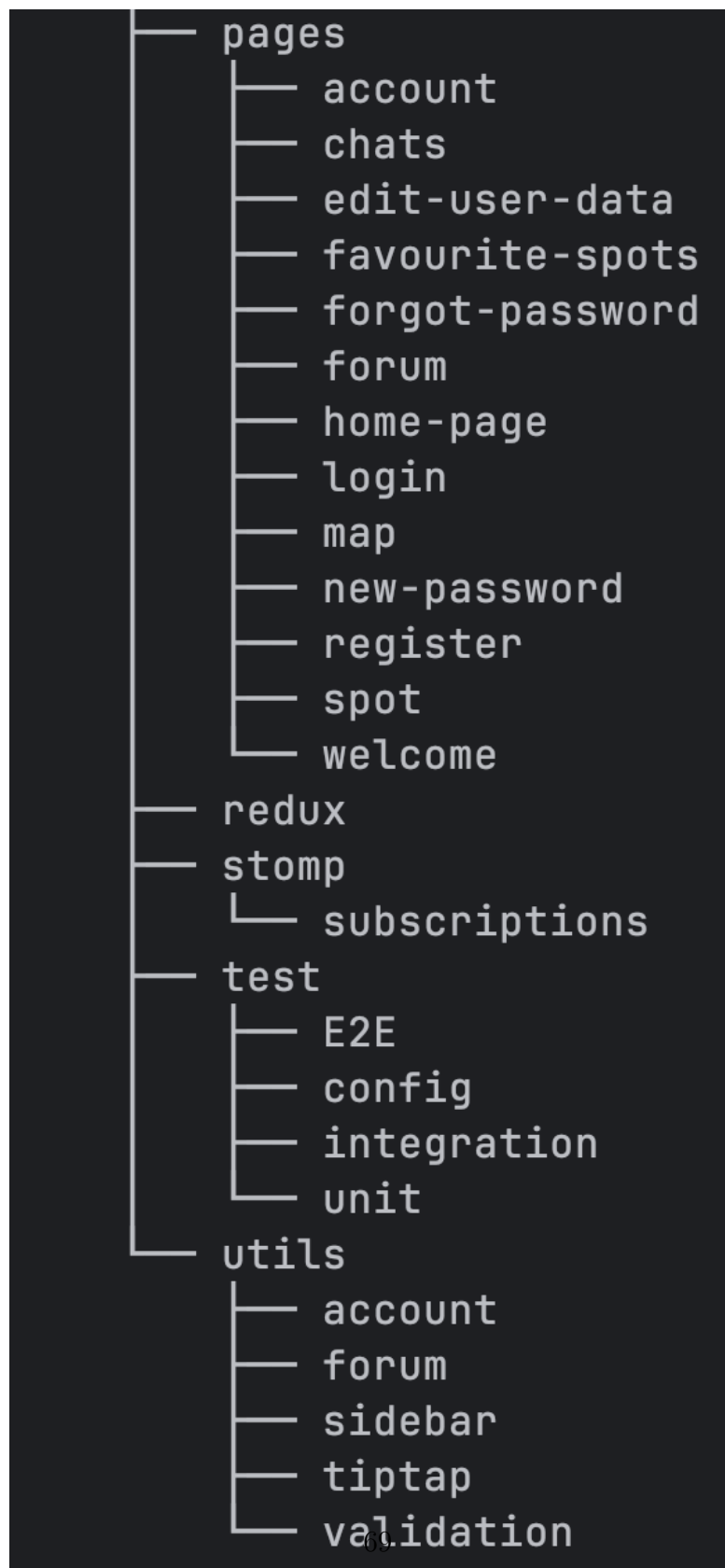
### 7.2 Implementacja frontendu

#### 7.2.1 Struktura aplikacji

Architektura aplikacji frontendowej została zaprojektowana 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 jest przedstawione na rysunkach [7.1](#) oraz [7.2](#).



Rysunek 7.1: Struktura katalogów (1)



Rysunek 7.2: 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 7.3: 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;

```

Rysunek 7.4: 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 [7.3](#) oraz [7.4](#)), wybrane

ścieżki zostały opakowane w komponent `ProtectedRoute`. Komponent ten pełni rolę bramki (rysunek 7.5).

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 }) {
  const isLoggedIn = useSelector((state) => state.account.isLoggedIn);

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

Rysunek 7.5: Implementacja komponentu bramki (`ProtectedRoute`)

### 7.2.2 Zarządzanie stanem i przepływ danych

W projekcie postawiliśmy na zrównoważone podejście do zarządzania [Stanem](#). Korzystamy zarówno z lokalnego [Stanu](#) komponentów (za pomocą [Hook \(React\)](#) `useState`) [15], jak i ze [Stanu](#) globalnego, utrzymywanego przez [Bibliotekę React Redux](#) [16]. Globalny [Stan](#) został wprowadzony po to, aby możliwie najbardziej ograniczyć przekazywanie [Propsów](#) w głąb drzewa komponentów oraz uniknąć niepotrzebnych ponownych renderów.

Do przechowywania [Stanu](#) lokalnego, ograniczonego tylko do danego komponentu (lub jego najbliższych elementów podrzędnych), wykorzystujemy [Hook \(React\)](#) `useState`. Natomiast efekty uboczne i synchronizację realizujemy za pomocą `useEffect`. W przypadku bardziej złożonej logiki lub potrzeby ponownego wykorzystania kodu powstały [Hook \(React\)](#)i niestandardowe, takie jak `useScreenSize`, `useDarkMode` czy `useClickOutside`. Dzięki temu większość logiki prezentacji została wydzielona z warstwy [UI](#), co poprawia czytelność i ułatwia utrzymanie kodu.

Z racji tego, że korzystamy z [Reacta](#) w połączeniu z [TypeScriptem](#), przygotowaliśmy również własne [Hook \(React\)](#)i wspomagające typowanie, takie jak `useDispatchTyped` oraz `useSelectorTyped`. Pozwalają one na bezpieczne typowanie

wanie akcji oraz selektorów [Reduxa](#) bez konieczności powtarzania adnotacji typów w każdym komponencie. Fragmenty tej implementacji przedstawiono na rysunkach [7.6](#) oraz [7.7](#).

```
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,
  },
});

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

Rysunek 7.6: 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, { setIsLoggedIn(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<{ setIsLoggedIn(state: W... = accountSlice.actions; Show usages  Mredosz

```

Rysunek 7.7: Przykładowy slice odpowiedzialny za sprawdzenie czy użytkownik jest zalogowany

### 7.2.3 Integracja i komunikacja z backendem

Jest to kluczowy element aplikacji, ponieważ wymaga bezpiecznego przesyłania danych użytkownika. W celu uproszczenia komunikacji z serwerem skorzystaliśmy z biblioteki [axios](#) [17] oraz [Biblioteki TanStack Query](#) [18]. We wszystkich ścieżkach, które wymagają aby użytkownik był zalogowany, do zapytania dołączany jest token [JWT](#). Token jest przekazywany w ciasteczku dzięki ustawieniu parametru `withCredentials` na wartość `true`. Przykładem pliku odpowiedzialnego za taką komunikację jest `account.js` (rys. 7.8 i 7.9), który obsługuje operacje związane z

logowaniem, rejestracją, zmianą hasła oraz wylogowaniem.

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

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

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

Rysunek 7.8: 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 7.9: Implementacja modułu `account` (2)

Funkcje odpowiedzialne za komunikację z backendem zostały umieszczone w katalogu `/http`. Dzięki temu są one scentralizowane i mogą być w prosty sposób wykorzystywane w różnych częściach aplikacji. Zastosowaliśmy TanStack Query, ponieważ znacząco ogranicza on powtarzalny kod oraz upraszcza obsługę błędów i stanów zapytania (takich jak ładowanie danych, błąd, sukces). [udostępniam.in](#) wartość `isLoading`, dzięki czemu komponent może łatwo wyświetlić ekran ładowania bez ręcznego zarządzania własnym stanem. Dodatkowo [Hook \(React\) useQuery](#) z tej [Biblioteki](#) umożliwia automatyczne pobieranie danych po wejściu na daną podstronę. Oznacza to, że komponent deklaruje jedynie „jakie dane są mu potrzebne”, a TanStack Query zajmuje się ich pobraniem, cache’owaniem oraz odświeżaniem. Do operacji, które wymagają wywołania akcji po stronie użytkownika (np. wysłania formularza logowania), wykorzystujemy [Hook \(React\) useMutation](#) z TanStack Query. Przykład użycia tego rozwiązania w procesie logowania został przedstawiony na rys. 7.10.
















```
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 7.10: Wykorzystanie TanStack Query przy logowaniu użytkownika

## 7.2.4 Style

Do stylowania interfejsu wykorzystaliśmy [Framework](#) Tailwind CSS [19]. Dzięki gotowym klasom udostępnianym przez Tailwind mogliśmy definiować wygląd elementów 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 nam również zachować spójność wizualną w całej aplikacji. W pliku `index.css` zdefiniowaliśmy zmienne kolorystyczne (rys. 7.11 i 7.12). Dzięki temu zmiana motywu kolorystycznego w przyszłości sprowadza się do edycji wartości w jednym miejscu.

	<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 7.11: Implementacja zmiennych kolorystycznych (1)





Rysunek 7.12: 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. Cała aplikacja

jest [Responsywna](#). Tailwind udostępnia predefiniowane prefiksy [Responsywne](#) (np. `md:`, `lg:`) (rys. 7.13), stworzyliśmy również własny (`3xl:`) na ekrany o rozdzielczości 2560px. Pozwalają one przypisywać style zależnie od szerokości ekranu bez pisania własnych reguł `@media`. 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 7.13: 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. 7.14).

```
<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 7.14: Przykładowe użycie klas Tailwind (w tym wariantu `dark:`)

Aby uzyskać płynniejsze i przyjemniejsze animacje, wykorzystaliśmy [Bibliotekę Motion](#) [20]. Pozwala ona w prosty sposób tworzyć animacje elementów interfejsu, bez potrzeby ręcznego pisania złożonych reguł [CSS](#). W naszej aplikacji użyliśmy jej `m.in.` w polach formularza logowania i rejestracji (rys. 7.15). 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.

```
<div className="relative">
  <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 7.15: Implementacja animacji z wykorzystaniem Motion

## 7.2.5 Wyszukiwarka spotów

Niniejszy rozdział opisuje sposób implementacji wyszukiwarki spotów.

Jednym z głównych modułów aplikacji jest wyszukiwarka spotów, która umożliwia użytkownikowi szybkie odnalezienie interesujących lokalizacji. Funkcjonuje ona w dwóch wariantach: prostym i zaawansowanym (rys. 7.16 oraz 7.17).

```

<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 7.16: 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 7.17: Implementacja zaawansowanej wersji wyszukiwarki

Przełączanie pomiędzy tymi widokami odbywa się za pomocą przycisku umieszczonego w górnej części strony (rys. 7.18).

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

Rysunek 7.18: Implementacja komponentu do przełączania trybów

W trybie prostym prezentowana jest karuzela (rys. 7.19) z dwunastoma najpopularniejszymi **spotami** w całej aplikacji. Użytkownik może w tym miejscu wyszukiwać **spoty** 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 7.19: Implementacja karuzeli z najpopularniejszymi [spotami](#)

Widok zaawansowany udostępnia rozszerzoną wyszukiwarę, która umożliwia filtrowanie wyników po mieście, tagach oraz ocenie, a także ich sortowanie według popularności i średniej oceny (rys. 7.17).

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. 7.18) zawiera dwa elementy `NavLink` z biblioteki `React Router`, co pozwala na przełączanie widoków bez konieczności przeładowywania całej strony.

W komponencie `SearchBar` (rys. 7.20) 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 użytkownik może wybrać interesującą go lokalizację, co ułatwia określenie, w jakich miejscach znajdują się dostępne [spoty](#).

```

<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 7.20: Implementacja prostej wyszukiwarki

Komponent `SearchSpotList` (rys. 7.21) 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 7.21: 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 lokalizacja [spota](#) jest prezentowana 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.. [15](#)

### 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](#), [13](#), [92](#)

### Backlog

Lista zadań, które należy wykonać w ramach projektu, używane w metodykach zwinnych.. [14](#)

### Biblioteka

Zewnętrzny lub wewnętrzny zestaw gotowych funkcji, klas, komponentów lub modułów, który można wielokrotnie wykorzystywać w projekcie zamiast pisać wszystko od zera. [70](#), [72](#), [74](#), [76](#), [79](#), [80](#), [85](#)

### BPMN

(ang. *Business Process Model and Notation*); standardowa notacja graficzna, która umożliwia szczegółowe przedstawienie i dokumentowanie procesów biznesowych.. [16](#)

## CI/CD

Skrót od *Continuous Integration/Continuous Deployment*. Praktyka programistyczna polegająca na automatyzacji procesu budowania, testowania i wdrażania oprogramowania. [14](#), [91](#), [94](#)

## 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.. [79](#), [80](#)

## Design

Etap lub proces projektowania wyglądu i funkcjonalności aplikacji, obejmujący zarówno aspekty wizualne, jak i użytkowe (UX/UI). [91](#)

## 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. . [9](#)

## Droniarz

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

## Droniarz foto/video

Pilot wykorzystujący drony fotograficzne/filmowe do rejestracji materiałów wizualnych (zdjęcia, wideo), zwykle z naciskiem na stabilizację i jakość obrazu.. [16](#)

## 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. [67](#)

## Framework

Zestaw narzędzi, bibliotek i struktur wspomagających tworzenie aplikacji. Ułatwia programowanie poprzez dostarczenie gotowych komponentów oraz określenie zasad organizacji kodu. [2](#), [77](#)

## Frontend

Warstwa aplikacji odpowiedzialna za interfejs użytkownika oraz interakcję z użytkownikiem. Zazwyczaj tworzona przy użyciu technologii takich jak HTML, CSS i JavaScript. [2](#), [13](#), [92](#)

## 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`).. [72](#), [76](#)

## IDE

(ang. *integrated development environment*); to zintegrowane środowisko programistyczne, służące do tworzenia, modyfikowania, testowania i konserwacji oprogramowania. [13](#)

## 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. [86](#)

## JWT

Skrót od *JSON Web Token*. Standard służący do bezpiecznego przekazywania informacji między stronami w formacie JSON, często używany w procesach autoryzacji użytkowników. [74](#), [91](#)

## 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*).. [100](#)

## 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](#)

## 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.. [72](#)

## 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). [71](#)

## React

Biblioteka JavaScript do budowy interfejsów użytkownika w oparciu o komponenty deklaratywne i wirtualny DOM. Zapewnia jednokierunkowy przepływ danych oraz zarządzanie stanem komponentów.. [72](#)

## 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.. [72](#), [73](#)

## 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.. [80](#)

## 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.

[14](#)

## Review kodu

Proces polegający na wzajemnym przeglądzie kodu źródłowego przez programistów w celu wykrycia błędów, poprawy jakości oraz zwiększenia spójności projektu. [14](#), [15](#), [91](#)

## Sidebar

Boczny panel w interfejsie użytkownika, zawierający menu nawigacyjne lub dodatkowe opcje funkcjonalne aplikacji. [54](#), [91–93](#)

## Spot

Spotkanie zespołu projektowego, zazwyczaj krótkie i regularne, służące omówieniu postępów prac, problemów oraz planów na najbliższy okres. [83–88](#)

## 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).. [72](#)

## 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”.. [14](#)

## 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.. [72](#)

## UI

Interfejs użytkownika (ang. *User Interface*); warstwa prezentacji odpowiedzialna za sposób wyświetlania danych oraz interakcji użytkownika z aplikacją.. [15](#), [72](#)

## UML

(ang. *Unified Modeling Language*); graficzny język wizualizacji, specyfikowania oraz dokumentowania składników systemów informatycznych. . [16](#)



# Spis tabel

2.1	Zespół projektowy . . . . .	7
2.2	Promotor . . . . .	8
2.3	Droniarze . . . . .	8
Tabela 3.1: Usługa zewnętrzna: GitHub Actions (CI) . . . . .		17
Tabela 3.2: Usługa zewnętrzna: Azure Blob Storage . . . . .		17
Tabela 3.3: Usługa zewnętrzna: Mailtrap . . . . .		17
Tabela 3.4: Usługa zewnętrzna: LocationIQ . . . . .		17
Tabela 3.5: Usługa zewnętrzna: Google Maps (Maps URLs) . . . . .		18
Tabela 3.6: Usługa zewnętrzna: OpenFreeMap . . . . .		18
Tabela 3.7: Usługa zewnętrzna: Open-Meteo . . . . .		18
Tabela 3.8: Usługa zewnętrzna: Tenor GIF API . . . . .		18
Tabela 3.9: Usługa zewnętrzna: Where the ISS at? . . . . .		19
Tabela 4.1: Karta wymagania ogólnego dla czatu: Wysyłanie wiadomości na czacie . . . . .		24
Tabela 4.2: Karta wymagania ogólnego dla czatu: Edycja czatu . . . . .		24
Tabela 4.3: Karta wymagania ogólnego dla czatu: Przeglądanie historii czatu . . . . .		24
Tabela 4.4: Karta wymagania ogólnego dla czatu: Tworzenie czatu . . . . .		25
Tabela 4.1: Wymaganie funkcjonalne dla czatu: Wysyłanie GIF-ów . . . . .		26
Tabela 4.2: Wymaganie funkcjonalne dla czatu: Wysyłanie plików . . . . .		27
Tabela 4.3: Wymaganie funkcjonalne dla czatu: Wysyłanie wiadomości prywatnych . . . . .		28

Tabela 4.4: Wymaganie funkcjonalne dla czatu: Wysyłanie wiadomości do wielu osób jednocześnie . . . . .	29
Tabela 4.5: Wymaganie funkcjonalne dla czatu: Rozpoczynanie nowego czatu . . . . .	30
Tabela 4.6: Wymaganie funkcjonalne dla czatu: Wysyłanie emotikonów .	31
Tabela 4.7: Wymaganie funkcjonalne dla czatu: Dostępność czatu po utworzeniu . . . . .	32
Tabela 4.8: Wymaganie funkcjonalne dla czatu: Edytowanie nazwy czatu grupowego . . . . .	33
Tabela 4.9: Wymaganie funkcjonalne dla czatu: Edycja zdjęcia czatu grupowego . . . . .	34
Tabela 4.10: Wymaganie funkcjonalne dla czatu: Edycja wysłanej wiadomości . . . . .	35
Tabela 4.11: Wymaganie funkcjonalne dla czatu: Usunięcie wysłanej wiadomości . . . . .	36
Tabela 4.12: Wymaganie funkcjonalne dla czatu: Dodawanie użytkowników do istniejącego czatu . . . . .	38
Tabela 4.13: Wymaganie funkcjonalne dla czatu: Wyświetlanie starszych wiadomości . . . . .	39
4.18 Profil użytkownika . . . . .	39
4.19 Lista dodanych spotów . . . . .	40
4.20 Dodanie spota . . . . .	41
4.21 Lista zdjęć . . . . .	42
4.22 Lista filmów . . . . .	42
4.23 Lista znajomych . . . . .	43
4.24 Lista obserwujących . . . . .	43
4.25 Lista obserwowanych . . . . .	44
4.26 Lista polubionych/odwiedzonych/planowanych spotów . . . . .	44
4.27 Lista komentarzy . . . . .	45
4.28 Ustawienia profilu . . . . .	46
4.29 Resetowanie hasła . . . . .	47

4.30 Dodawanie do znajomych . . . . .	48
4.31 Logowanie i rejestracja . . . . .	49
4.32 Strona główna — podstawowe filtry . . . . .	50
4.33 Strona główna — zaawansowane filtry . . . . .	51
4.34 Ustawienia motywu (ręczna zmiana) . . . . .	52
4.35 Zapamiętanie preferencji motywu . . . . .	53
4.36 Szybki przełącznik motywu w interfejsie . . . . .	54
Tabela 4.1: Wymaganie pozafunkcjonalne dla czatu: Ograniczenie wi- doczności czatów do członków . . . . .	56
Tabela 4.2: Wymaganie pozafunkcjonalne dla czatu: Wymóg zalogowania do korzystania z czatu . . . . .	57
Tabela 4.3: Wymaganie pozafunkcjonalne dla czatu: Grupowanie wiado- mości według daty wysłania . . . . .	58
Tabela 4.4: Wymaganie pozafunkcjonalne dla czatu: Wyraźne oznaczenie nadawcy i czasu wysłania . . . . .	59
Tabela 4.5: Wymaganie pozafunkcjonalne dla czatu: Czas załadowania starszych wiadomości poniżej 3 sekund . . . . .	61
Tabela 4.6: Wymaganie pozafunkcjonalne dla czatu: Natychmiastowe wy- syłanie wiadomości . . . . .	62
Tabela 4.7: Wymaganie pozafunkcjonalne dla czatu: Zachowanie wiado- mości przy chwilowej utracie połączenia . . . . .	63
Tabela 4.8: Wymaganie pozafunkcjonalne dla czatu: Limit wysyłanych wiadomości w jednostce czasu . . . . .	64

# Bibliografia

- [1] *Disciplined Agile Delivery*. PMI. 1 stycznia 2025. URL: <https://www.pmi.org/disciplined-agile/process/introduction-to-dad/why> (dostęp 30.10.2025).
- [2] *Disciplined Agile Delivery — Lean Life Cycle*. PMI. 1 stycznia 2025. URL: <https://www.pmi.org/disciplined-agile/lifecycle/lean-lifecycle> (dostęp 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 maja 2024. URL: <https://wolski.pro/2024/05/10-wskazowek-poprawiajacych-modelowanie-procesow-biznesowych-w-notacji-bpmn/> (dostęp 19.11.2025).
- [5] *About billing for GitHub Actions*. GitHub Docs. 1 stycznia 2024. URL: <https://docs.github.com/en/billing/managing-billing-for-github-actions/about-billing-for-github-actions> (dostęp 2.11.2025).
- [6] *Scalability and performance targets for Blob storage*. Microsoft Learn. 1 stycznia 2024. URL: <https://learn.microsoft.com/azure/storage/blobs/scalability-targets> (dostęp 2.11.2025).
- [7] *What are the limitations in Mailtrap?* Mailtrap Docs. 1 stycznia 2024. URL: <https://help.mailtrap.io/article/111-what-are-the-limitations-in-mailtrap/> (dostęp 2.11.2025).
- [8] *LocationIQ Pricing*. LocationIQ. 1 stycznia 2024. URL: <https://locationiq.com/pricing> (dostęp 2.11.2025).
- [9] *Google Maps (Maps URLs)*. Google Maps. 1 stycznia 2024. URL: <https://developers.google.com/maps/documentation/urls/get-started?hl=pl> (dostęp 2.11.2025).
- [10] *OpenFreeMap Documentation*. OpenFreeMap. 1 stycznia 2024. URL: <https://openfreemap.org/docs> (dostęp 2.11.2025).

- [11] *OpenFreeMap Quick Start*. OpenFreeMap. 1 stycznia 2024. URL: <https://openfreemap.org/docs/quick-start> (dostęp 2.11.2025).
- [12] *Open-Meteo API Usage & Pricing*. Open-Meteo. 1 stycznia 2024. URL: <https://open-meteo.com/en/docs/usage-and-pricing> (dostęp 2.11.2025).
- [13] *Tenor API — Documentation*. Tenor. 1 stycznia 2024. URL: <https://tenor.com/gifapi/documentation> (dostęp 2.11.2025).
- [14] *Where the ISS at? API*. wheretheiss.at. 1 stycznia 2024. URL: <https://wheretheiss.at/> (dostęp 2.11.2025).
- [15] *React useState*. 1 stycznia 2025. URL: <https://react.dev/reference/react/useState> (dostęp 3.11.2025).
- [16] *Redux*. 1 stycznia 2025. URL: <https://redux.js.org/> (dostęp 3.11.2025).
- [17] *Axios*. 1 stycznia 2025. URL: <https://axios-http.com/> (dostęp 3.11.2025).
- [18] *Tanstack Query*. 1 stycznia 2025. URL: <https://tanstack.com/query/latest> (dostęp 3.11.2025).
- [19] *Tailwind*. 1 stycznia 2025. URL: <https://tailwindcss.com/> (dostęp 3.11.2025).
- [20] *Motion*. 1 stycznia 2025. URL: <https://motion.dev/> (dostęp 3.11.2025).

# 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.