



POLSKO-JAPONSKA 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: spotty-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 stylów 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

| | |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------|
| Temat projektu: Aplikacja webowa: spoty-na-drony.pl Temat projektu po angielsku: Web application: spoty-na-drony.pl | Akronim: Merkury Data ustalenia tematu 2023-10-10 |
| Promotor: mgr Adam Urbanowicz | Konsultanci: 1. — brak — |
| Cele projektu: Stworzenie w pełni funkcjonalnej aplikacji internetowej do rozwijania hobby(latania dronem). | |
| Rezultaty projektu: 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. | |
| Miary sukcesu: Gotowa do wdrożenia aplikacja. Realizacja w terminie zgodnym z wymaganiami. | |
| Ograniczenia: Budżetowe: brak środków na wdrożenie. Zawodowe: brak doświadczenia. Czasowe: trzy semestry (09.2024 - 02.2026). Ludzkie: czteroosobowy zespół. | |

| Wykonawcy | Numer albumu | 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 |

| | |
|-----------------------------------------------------|------------------------------------------------------|
| Data ukończenia projektu: 16 grudnia 2025 | Recenzent: dr Elżbieta Puźniakowska-Gałuch |
|-----------------------------------------------------|------------------------------------------------------|

Spis treści

| | |
|------------------------------------------------------------------------------------|-----------|
| 1 Wstęp | 7 |
| 1.1 O projekcie | 7 |
| 1.2 Cel i zakres prac | 7 |
| 1.3 Geneza pomysłu | 7 |
| 2 Opis problemu | 8 |
| 2.1 Rich picture | 8 |
| 2.2 Udziałowcy | 8 |
| 2.3 Istniejące rozwiązania | 10 |
| 2.4 Wizja rozwiązania | 10 |
| 2.5 Aspekty społeczne i biznesowe | 10 |
| 2.5.1 Aspekty społeczne | 10 |
| 2.5.2 Aspekty biznesowe | 10 |
| 3 Planowanie | 11 |
| 3.1 Metodologia pracy | 11 |
| 3.1.1 Przegląd rozważanych podejść | 11 |
| 3.1.2 Odrzucone podejścia | 11 |
| 3.1.3 Wybrane podejście: Disciplined Agile Delivery (Lean Life Cycle) | 12 |
| 3.1.4 Narzędzia i komunikacja | 12 |
| 3.1.5 Podział ról w zespole | 13 |
| 3.2 Harmonogram projektu | 13 |
| 3.3 Technologie i narzędzia | 15 |
| 3.3.1 Technologie | 15 |

| | | |
|----------|-------------------------------------------------------|------------|
| 3.3.2 | Narzędzia | 24 |
| 3.4 | Zasoby i ograniczenia | 27 |
| 3.4.1 | Zasoby | 27 |
| 3.4.2 | Ograniczenia | 27 |
| 3.4.3 | Usługi zewnętrzne | 27 |
| 3.5 | Analiza ryzyka | 30 |
| 4 | Analiza wymagań | 31 |
| 4.1 | Przypadki użycia | 32 |
| 4.1.1 | Aktorzy | 32 |
| 4.1.2 | Diagramy przypadków użycia | 34 |
| 4.1.3 | Scenariusze przypadków użycia | 42 |
| 4.2 | Wymagania ogólne i dziedzinowe | 84 |
| 4.3 | Wymagania funkcjonalne | 84 |
| 4.3.1 | Funkcjonalności dla mapy | 84 |
| 4.3.2 | Funkcjonalności dla chatu | 84 |
| 4.3.3 | Funkcjonalności dla forum | 84 |
| 4.3.4 | Funkcjonalności dla konta użytkownika | 84 |
| 4.3.5 | Funkcjonalności dla logowania i rejestracji | 94 |
| 4.3.6 | Funkcjonalności dla wyszukiwarki spotów | 95 |
| 4.3.7 | Funkcjonalności dla motywu | 97 |
| 4.4 | Wymagania pozafunkcjonalne | 99 |
| 4.5 | Wymagania interfejs z otoczeniem | 99 |
| 4.6 | Wymagania na środowisko docelowe | 99 |
| 5 | Projekt | 100 |
| 5.1 | Wzorce projektowe | 100 |
| 5.2 | Architektura systemu | 100 |
| 5.2.1 | Diagram architektury | 101 |
| 5.2.2 | Komponenty systemu | 103 |
| 5.3 | Projekt bazy danych | 104 |
| 5.3.1 | Model danych | 104 |

| | | |
|----------|------------------------------------------------|------------|
| 5.3.2 | Diagram ERD | 104 |
| 5.4 | Architektura interfejsu użytkownika | 104 |
| 5.4.1 | Projekt strony głównej | 104 |
| 5.4.2 | Projekt panelu logowania | 104 |
| 5.4.3 | Projekt mapy | 104 |
| 5.4.4 | Projekt chatu | 104 |
| 5.4.5 | Projekt forum | 104 |
| 5.4.6 | Projekt konta użytkownika | 104 |
| 6 | Przebieg realizacji projektu | 105 |
| 6.1 | Sprint 1 | 105 |
| 6.2 | Sprint 2 | 105 |
| 7 | Realizacja Projektu | 106 |
| 7.1 | Implementacja backendu | 106 |
| 7.1.1 | Struktura projektu | 106 |
| 7.1.2 | Endpointy systemu | 109 |
| 7.1.3 | Integracja z bazą danych | 155 |
| 7.1.4 | Obsługa uwierzytelnienia | 158 |
| 7.1.5 | Konteneryzacja | 158 |
| 7.2 | Implementacja frontendu | 158 |
| 7.2.1 | Struktura aplikacji | 158 |
| 7.2.2 | Zarządzanie stanem i przepływ danych | 163 |
| 7.2.3 | Integracja i komunikacja z backendem | 166 |
| 7.2.4 | Style | 169 |
| 7.2.5 | Wyszukiwarka spotów | 173 |
| 7.2.6 | Mapa | 180 |
| 7.2.7 | Chat | 180 |
| 7.2.8 | Forum | 180 |
| 7.2.9 | Konto użytkownika | 180 |
| 7.2.10 | Panel logowania | 180 |
| 7.3 | Implementacja CI/CD | 180 |

| | | |
|-----------|-----------------------------------------------------|------------|
| 7.4 | Zastosowanie WebSocket | 180 |
| 7.4.1 | Charakterystyka protokołu WebSocket | 180 |
| 7.4.2 | Zastosowanie WebSocket w naszym projekcie | 181 |
| 7.4.3 | Implementacja na backendzie | 182 |
| 7.4.4 | Implementacja na frontendzie | 187 |
| 7.4.5 | Przebieg komunikacji | 199 |
| 8 | Testy | 200 |
| 8.1 | Testy jednostkowe | 200 |
| 8.2 | Testy integracyjne | 200 |
| 8.3 | Testy E2E | 200 |
| 8.4 | Wyniki testów i wnioski | 200 |
| 9 | Prezentacja systemu | 201 |
| 9.1 | Strona główna | 201 |
| 9.2 | Strona mapy | 201 |
| 9.3 | Strona chatu | 201 |
| 9.4 | Strona forum | 201 |
| 9.5 | Panel logowania | 201 |
| 9.6 | Panel konta użytkownika | 201 |
| 10 | Nakład pracy | 202 |
| 10.1 | Ogólny nakład pracy | 202 |
| 10.2 | Indywidualne nakłady pracy | 202 |
| 10.2.1 | Adam Langmesser | 202 |
| 10.2.2 | Mateusz Redosz | 202 |
| 10.2.3 | Stanisław Oziemczuk | 205 |
| 10.2.4 | Kacper Badek | 205 |
| 11 | Podsumowanie | 206 |
| 11.1 | Osiągnięte rezultaty | 206 |
| 11.2 | Napotkane wyzwania | 206 |
| 11.3 | Plany na przyszłość | 206 |

| | |
|-----------------------------------|------------|
| 12 Słownik pojęć i skrótów | 207 |
| Spis tabel | 221 |
| Bibliografia | 227 |
| Załączniki | 229 |

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

Tabela 2.1: Karta udziałowca: Zespół projektowy

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

Tabela 2.2: Karta udziałowca: Promotor

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

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

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

2.3 Istniejące rozwiązania

2.4 Wizja rozwiązania

2.5 Aspekty społeczne i biznesowe

2.5.1 Aspekty społeczne

2.5.2 Aspekty biznesowe

Rozdział 3

Planowanie

3.1 Metodologia pracy

3.1.1 Przegląd rozważanych podejść

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

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

3.1.2 Odrzucone podejścia

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

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

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

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

Podjęto decyzję o zastosowaniu **Disciplined Agile Delivery** [1] w wariantie **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 rôle 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

W ramach prac nad projektem 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

Do realizacji projektu zespół wspólnie wytypował główne technologie części [backendowej](#), [frontendowej](#) oraz dokumentacji. Natomiast poszczególne biblioteki i rozwiązania były wybierane indywidualnie lub po konsultacjach przez osobę wykonującą dane zadanie. Poniżej przedstawiono stos technologiczny zastosowany w projekcie.

• [Backend](#)

Na główny [framework](#) został wybrany SpringBoot, ponieważ spośród innych dostępnych opcji, członkowie zespołu mieli z nim największe doświadczenie

nabyte zarówno podczas studiów, jak i później pracę komercyjną. Językiem programowania wykorzystywanym w SpringBoot'cie jest Java, z którym zespół zapoznał się w ramach programu nauczania.

- **Java** – obiektowy język programowania, cechujący się silnym typowaniem. Programy napisane w Javie są uruchamiane na maszynie wirtualnej Java ([JVM](#)), dzięki czemu można je bezproblemowo przenosić między różnymi platformami wyposażonymi w to środowisko.
- **SpringBoot** – [framework](#) służący do tworzenia aplikacji opartych na [Spring Framework](#). Wykorzystuje strategię [Convention Over Configuration](#), która zmniejsza czas na konfigurowanie Springa, pozwalając skupić się na implementacji logiki. Służy do tworzenia między innymi aplikacji internetowych czy mikroserwisów.

| Zestawienie używanych bibliotek na backendzie | |
|-----------------------------------------------|----------------------------------------------------------------------------------------------|
| Biblioteka | Opis |
| angus-mail | Wysyłanie i odbiór wiadomości e-mail w aplikacjach Java. |
| azure-storage-blob | Operacje na blobach w Microsoft Azure Blob Storage, np. upload, download. |
| GeographicLib-Java | Precyjne obliczenia geodezyjne i konwersja współrzędnych. |
| h2 | Lekka baza danych H2 uruchamiana w pamięci RAM używana w testach jako zastępstwo prawdziwej. |
| httpclient5 | Zaawansowany klient HTTP do wykonywania żądań i obsługi odpowiedzi. |
| httpcore5 | Niskopoziomowe elementy HTTP wykorzystywane przez httpclient. |
| jjwt-api | Tworzenie i parsowanie JWT. |
| jjwt-impl | Implementacja funkcjonalna biblioteki JJWT. |

| Biblioteka | Opis |
|--------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| jjwt-jackson | Integracja JWT z Jacksonem dla serializacji i deserializacji zawartości JWT. |
| jsoup | Parsowanie, manipulacja i ekstrakcja danych z dokumentów HTML. |
| junit-jupiter | Integracja biblioteki Testcontainers z frameworkiem testowym JUnit ułatwiająca pisanie testów integracyjnych z użyciem kontenerów. |
| lombok | Biblioteka ułatwiająca generowanie kodu (getters, setters, buildery, konstruktor itp.) przez adnotacje zmniejszające ilość boilerplate'u w kodzie. |
| postgresql | Umożliwia połączenie i komunikację z bazą danych PostgreSQL. |
| shedlock-spring | Zarządzanie zadaniami okresowymi (cron/scheduled). |
| spring-boot-starter-websocket | Wsparcie Spring Boot dla komunikacji WebSocket – konfiguracja i zależności umożliwiające dwukierunkową komunikację w czasie rzeczywistym w aplikacjach webowych. |
| spring-security-messaging | Integracja Spring Security z warstwą messaging (STOMP/WebSocket) – uwierzytelnianie i autoryzacja komunikatów. |
| spring-boot-starter-cache | Abstrakcje i automatyczna konfiguracja cache w Spring Boot ułatwiające włączenie mechanizmów cache'owania. |
| spring-boot-starter-data-redis | Integracja Spring Data Redis dodające klienta, repozytoria i konfiguracje do współpracy z Redis. |
| spring-boot-starter-data-jpa | Warstwa dostępu do relacyjnej bazy danych przez JPA. |
| spring-boot-starter-web | Podstawowy starter webowy Spring Boot: Spring MVC, Jackson, wbudowany serwer aplikacyjny dla REST/HTTP. |

| Biblioteka | Opis |
|--------------------------------------------|----------------------------------------------------------------------------------------------------------------|
| spring-boot-starter-validation | Wsparcie walidacji Bean Validation (Jakarta Validation / Hibernate Validator) dla danych wejściowych. |
| spring-boot-starter-aop | Obsługa programowania aspektowego (AOP) w Springu – aspekty, przechwytywanie wywołań, transakcyjne zachowania. |
| spring-boot-starter-actuator | Monitoring, zbieranie metryk aplikacji Spring Boot. |
| spring-boot-starter-oauth2-resource-server | Wsparcie serwera zasobów OAuth2 w Spring Boot – walidacja tokenów i konfiguracja zabezpieczeń zasobów. |
| spring-boot-configuration-processor | Procesor adnotacji dla konfiguracji Spring Boot – generacja metadanych dla właściwości konfiguracyjnych. |
| spring-boot-starter-test | Zestaw narzędzi testowych (JUnit, Mockito, AssertJ itp.) do testów jednostkowych i integracyjnych. |
| spring-security-test | Narzędzia pomocnicze i rozszerzenia do testowania konfiguracji Spring Security. |
| spring-boot-starter-oauth2-client | Wsparcie klienta OAuth2 w Spring Boot – logowanie/połączenie z zewnętrznymi providerami OAuth2/OIDC. |
| spring-boot-starter-security | Podstawowe komponenty Spring Security do zabezpieczania aplikacji. |
| spring-boot-starter-webflux | Budowanie reaktywnych (asynchronicznych / nieblokujących) aplikacji webowych. |
| spring-retry | Biblioteka do automatycznego ponawiania operacji z konfiguracją i adnotacjami. |
| spring-boot-starter-thymeleaf | Integracja Thymeleaf z Spring Boot – silnik szablonów do generowania widoków HTML po stronie serwera. |
| testcontainers | Uruchamianie izolowanych kontenerów Docker w testach integracyjnych. |

Tabela 3.1: Zestawienie wszystkich bibliotek użytych na backendzie.

• **Frontend**

Do realizacji tej części projektu wybrano bibliotekę React JavaScript, która wszyscy członkowie zespołu poznali w trakcie studiów w ramach wybranej specjalizacji Aplikacje Internetowe.

- **React** – biblioteka JavaScript służąca do budowania interaktywnych interfejsów użytkownika (**UI**). Polega na programowaniu deklaratywnym oraz tworzeniu komponentów wielokrotnego użytku. Nie manipuluje bezpośrednio **DOM**, lecz tworzy swój wirtualny **DOM** i porównuje jego wersje. Po wykryciu zmian aktualizuje tylko te części **DOM**, które tego wymagają, co przekłada się na wydajną interakcję z aplikacją. Często jest wykorzystywany do tworzenia aplikacji typu **SPA**.

| Zestawienie używanych bibliotek na frontendzie | |
|------------------------------------------------|---------------------------------------------------------------------------------------------------------------|
| Biblioteka / plugin | Opis / przeznaczenie |
| @ferrucc-io/emoji-picker | Komponent umożliwiający wybór emotikon w aplikacji. |
| @hookform/resolvers | Adaptery validatorów dla react-hook-form – ułatwia integrację z bibliotekami walidacji (Zod, Yup). |
| @reduxjs/toolkit | Narzędzie upraszczające konfigurację i używanie Reduxa. |
| @stomp/stompjs | Klient protokołu STOMP do komunikacji poprzez WebSocket obsługujący sesje, subskrypcje i wymiany komunikatów. |
| @tailwindcss/vite | Wtyczka integrująca Tailwind CSS z bundlerem Vite. |
| @tanstack/react-query | Zarządzanie asynchronicznymi danymi: pobieranie, cache'owanie, synchronizacja i obsługa błędów zapytań HTTP. |

| Biblioteka / plugin | Opis / przeznaczenie |
|--------------------------------|------------------------------------------------------------------------------------------------------------|
| @tiptap/extension-file-handler | Rozszerzenie edytora Tiptap dodające obsługę wstawiania i zarządzania plikami. |
| @tiptap/extension-image | Rozszerzenie Tiptap pozwalające na wstawianie i obsługę obrazów w edytorze. |
| @tiptap/extension-placeholder | Rozszerzenie Tiptap dodające placeholder w polach edytora. |
| @tiptap/extension-text-align | Rozszerzenie Tiptap umożliwiające wyrównywanie tekstu. |
| @tiptap/pm | Silnik edytora (ProseMirror) używany wewnętrznie przez Tiptap obsługujący model dokumentu i transformacji. |
| @tiptap/react | Integracja edytora Tiptap z Reactem dodająca komponenty i hooki edytora. |
| @tiptap/starter-kit | Podstawowe rozszerzenia i konfiguracja Tiptap. |
| @vis.gl/react-maplibre | Narzędzia do implementacji map. |
| antd | Komponenty UI dla React o ustandaryzowanym wyglądzie. |
| axios | Obiektowy klient HTTP służący do wykonywania zapytań oraz obsługi odpowiedzi i błędów. |
| date-fns | Funkcje do manipulacji i formatowania dat. |
| dotenv | Ładowanie zmiennych środowiskowych z pliku .env. |
| maplibre-gl | Silnik renderowania map – warstwy, kafelki i interakcje geograficzne. |
| media-chrome | Elementy UI obsługujące odtwarzanie multimedialów w przeglądarce. |
| motion | Narzędzia służące do animacji interfejsu użytkownika. |

| Biblioteka / plugin | Opis / przeznaczenie |
|-----------------------------|-----------------------------------------------------------------------------------------------|
| query-string | Narzędzia do parsowania i generowania parametrów zapytań w URL. |
| react | Biblioteka do budowy deklaratywnych interfejsów użytkownika oparta na komponentach. |
| react-dom | Pakiet odpowiedzialny za renderowanie elementów React w przeglądarkowym DOM. |
| react-hook-form | Obsługa formularzy: zarządzanie stanem, walidacje i wydajność. |
| react-icons | Zbiór ikon udostępnionych jako komponenty React. |
| react-intersection-observer | Obserwowanie wejścia/wyjścia elementów z pola widzenia. |
| react-paginate | Komponent pomocniczy do paginacji list i tabel w aplikacji React. |
| react-player | Odtwarzanie multimedialnych w aplikacji. |
| react-redux | Integracja Redux i Reacta. |
| react-router-dom | Routing w aplikacjach React. |
| react-select | Rozszerzony komponent <code>select</code> z opcjami wyszukiwania, grupowania i stylizacji. |
| sockjs-client | Klient WebSocket z fallbackami umożliwiający stabilniejszą komunikację w czasie rzeczywistym. |
| tailwindcss | Definiowanie wyglądu przy pomocy gotowych klas. |
| uuid | Generator unikalnych identyfikatorów. |
| victory | Tworzenie wykresów i wizualizacji danych w React. |
| victory-chart | Moduł biblioteki Victory zawierający komponenty i narzędzia do rysowania wykresów. |
| zod | Typowana walidacja danych. |
| @testing-library/jest-dom | Rozszerzenia matcherów dla Jesta do asercji DOM. |

| Biblioteka / plugin | Opis / przeznaczenie |
|----------------------------------|-------------------------------------------------------------------------------------|
| @testing-library/react | Narzędzia do testowania komponentów React. |
| @testing-library/user-event | Symulacja zdarzeń użytkownika (np. kliknięcie) w testach integracyjnych. |
| @types/react | Typy TypeScript dla biblioteki React. |
| @types/react-dom | Typy TypeScript dla react-dom. |
| @types/sockjs-client | Typy TypeScript dla klienta SockJS. |
| @typescript-eslint/eslint-plugin | Zestaw reguł ESLint specyficznych dla TypeScript. |
| @typescript-eslint/parser | Parser ESLint umożliwiający analizę kodu TypeScript. |
| @vitejs/plugin-react | Wtyczka Vite zapewniająca obsługę JSX/React Fast Refresh i optymalizacje. |
| cypress | Narzędzie do testów end-to-end. |
| eslint | Narzędzie do analizy statycznej kodu na podstawie zdefiniowanych reguł. |
| eslint-plugin-react | Wtyczka ESLint z regułami dedykowanymi dla aplikacji React. |
| eslint-plugin-react-hooks | Reguły ESLint dotyczące hooków React. |
| eslint-plugin-react-refresh | Wtyczka wspomagająca integrację z React Fast Refresh. |
| jest | Tworzenie testów jednostkowych JavaScript. |
| jsdoc | Narzędzie do generowania dokumentacji API z adnotacjami w kodzie źródłowym. |
| jsdom | Implementacja DOM w Node.js używana w testach do symulacji środowiska przeglądarki. |
| prettier | Formatowanie i ujednolicenie stylu kodu. |

| Biblioteka / plugin | Opis / przeznaczenie |
|-----------------------------|---------------------------------------------------------------------------------|
| prettier-plugin-tailwindcss | Wtyczka Prettiera sortująca klasy Tailwind CSS. |
| tailwind-scrollbar | Plugin Tailwind CSS dodający klasy pomocnicze do stylizacji pasków przewijania. |
| typescript | Nakładka JavaScript z systemem typów. |
| vite | Bundler zoptymalizowany pod nowoczesne aplikacje webowe. |
| vitest | Narzędzie do uruchamiania testów zoptymalizowane pod bundler Vite. |

Tabela 3.2: Zestawienie wszystkich bibliotek i pluginów użytych na frontendzie.

- **Cache**

- **Redis** – z ang. REmote DIctionary Server, nierelacyjna (NoSQL) baza danych przechowującą dane jako pary klucz - wartość. Działa w pamięci RAM, dzięki czemu pozwala na bardzo szybki odczyt danych.

- **Konteneryzacja**

- **Docker** – to silnik do konteneryzacji oprogramowania. Tworzy środowisko oddzielone od systemu hosta, w którym na podstawie obrazów programów tworzone są ich kontenery, czyli niezależne ich instancje. Docker pobiera wszystkie niezbędne dependencje dla danego kontenera, bez potrzeby uruchamiania osobnego systemu operacyjnego, dzięki czemu kontenery są znacznie lżejsze niż maszyny wirtualne. Konteneryzacja pozwala na uruchomienie oprogramowania na różnych komputerach dokładnie w taki sam sposób, rozwiązuje to problem „na mojej maszynie działa”. Członkowie zespołu zapoznali się tą z technologią w trakcie realizacji specjalizacji Aplikacje Internetowe.

- **Baza danych**

- **PostgreSQL** – system zarządzania relacyjnymi bazami danych, zgodny ze standardem SQL. Wybrano relacyjną bazę danych, ponieważ doskonale wpisuje się ona w planowaną strukturę danych.

3.3.2 Narzędzia

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

- **IntelliJ IDEA Ultimate**

Jest to [IDE](#) od firmy JetBrains. Dzięki licznie dostępnym pluginom oferuje obsługę wielu języków programowania oraz innych składni. Pozwala również na integrację z repozytorium. Używano go do programowania zarówno [frontendu](#), jak i [backendu](#) oraz tworzenia dokumentacji w LaTeX.

- **Docker Desktop**

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

- **Docker Compose**

Narzędzie, które pozwala definiować oraz uruchamiać aplikacje składające się z wielu kontenerów Docker. Konfiguracja serwisów, sieci i [wolumenów](#) jest ustawiana w pliku (lub plikach) YAML. Zastosowano je do skonfigurowania bazy danych i serwisu do [cache'owania](#) w środowisku deweloperskim.

- **One Drive**

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

- **Azure Blob Storage**

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

- **Jira**

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

- **GitHub**

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

- **GitHub Actions**

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

- **GitHub Copilot**

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

- **Discord**

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

- **Messenger**

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

- **Postman**

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

- **Figma**

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

- **Visual Paradigm**

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

- **Xmind**

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

3.4 Zasoby i ograniczenia

3.4.1 Zasoby

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

3.4.2 Ograniczenia

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

3.4.3 Usługi zewnętrzne

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

| KARTA USŁUGI ZEWNĘTRZNEJ | |
|--------------------------|---------------------------------------------------------|
| Identyfikator: | UZ3 |
| Nazwa: | GitHub Actions (CI) [5] |
| Opis: | Uruchomienia pipeline'ów CI/CD dla repozytorium GitHub. |
| Limit: | 3000 min/mies. |

Tabela 3.3: Usługa zewnętrzna: GitHub Actions (CI)

| KARTA USŁUGI ZEWNĘTRZNEJ | |
|--------------------------|------------------------------------------------------------|
| Identyfikator: | UZ4 |
| Nazwa: | Azure Blob Storage [6] |
| Opis: | Magazyn plików (m.in. zdjęcia spotów, załączniki z czatu). |
| Limit: | 1 GB/mies. |

Tabela 3.4: Usługa zewnętrzna: Azure Blob Storage

| KARTA USŁUGI ZEWNĘTRZNEJ | |
|--------------------------|----------------------------------------------------------|
| Identyfikator: | UZ5 |
| Nazwa: | Mailtrap [7] |
| Opis: | Środowisko testowe SMTP oraz Email API do wysyłki maili. |
| Limit: | 150 maili/dzień |

Tabela 3.5: Usługa zewnętrzna: Mailtrap

| KARTA USŁUGI ZEWNĘTRZNEJ | |
|--------------------------|---------------------------------------------------|
| Identyfikator: | UZ6 |
| Nazwa: | LocationIQ [8] |
| Opis: | Geokodowanie adresu przy dodawaniu nowych spotów. |
| Limit: | 5 000 zapytań/dzień |

Tabela 3.6: Usługa zewnętrzna: LocationIQ

| KARTA USŁUGI ZEWNĘTRZNEJ | |
|--------------------------|--------------------------------------------------------------|
| Identyfikator: | UZ7 |
| Nazwa: | Google Maps (Maps URLs) [9] |
| Opis: | Otwieranie nawigacji w aplikacji Map Google (deep link/URL). |
| Limit: | Brak limitu w ramach dokumentowanego sposobu użycia. |

Tabela 3.7: Usługa zewnętrzna: Google Maps (Maps URLs)

| KARTA USŁUGI ZEWNĘTRZNEJ | |
|--------------------------|-------------------------------------------------------|
| Identyfikator: | UZ8 |
| Nazwa: | OpenFreeMap [10] |
| Opis: | Publiczny serwer kafelków do renderu mapy na froncie. |
| Limit: | 30 000 zapytań/mies. |

Tabela 3.8: Usługa zewnętrzna: OpenFreeMap

| KARTA USŁUGI ZEWNĘTRZNEJ | |
|--------------------------|-----------------------------------------|
| Identyfikator: | UZ9 |
| Nazwa: | Open-Meteo [11] |
| Opis: | Prognozy pogody wyświetlane dla spotów. |
| Limit: | 10 000 zapytań/dzień |

Tabela 3.9: Usługa zewnętrzna: Open-Meteo

| KARTA USŁUGI ZEWNĘTRZNEJ | |
|--------------------------|---------------------------------------------------------|
| Identyfikator: | UZ10 |
| Nazwa: | Tenor GIF API [12] |
| Opis: | Wyszukiwanie GIF-ów w czacie. |
| Limit: | 1 zapytanie na sekundę; brak ogólnego limitu dziennego. |

Tabela 3.10: Usługa zewnętrzna: Tenor GIF API

| KARTA USŁUGI ZEWNĘTRZNEJ | |
|--------------------------|----------------------------------------------------------|
| Identyfikator: | UZ11 |
| Nazwa: | Where the ISS at? [13] |
| Opis: | HTTP API z bieżącą pozycją satelity, używane pomocniczo. |
| Limit: | 1 zapytanie na sekundę; brak ogólnego limitu dziennego. |

Tabela 3.11: Usługa zewnętrzna: Where the ISS at?

3.5 Analiza ryzyka

Rozdział 4

Analiza wymagań

W niniejszym rozdziale przedstawiono analizę wymagań stawianych aplikacji. Punktem wyjścia była identyfikacja i opracowanie przypadków użycia systemu (zarówno w postaci diagramów, jak i szczegółowych scenariuszy), które stanowią podstawę do dalszej specyfikacji zachowania systemu.

Na tej podstawie zdefiniowano kolejno:

- wymagania ogólne oraz dziedzinowe,
- wymagania funkcjonalne,
- wymagania pozafunkcjonalne,
- wymagania dotyczące interfejsu z otoczeniem,
- wymagania na środowisko docelowe.

Do określania priorytetów realizacji poszczególnych wymagań wykorzystano technikę [MoSCoW](#). Metoda ta wyróżnia cztery kategorie:

M (Must have) wymagania kluczowe, które muszą zostać zrealizowane, aby system mógł zostać uznany za działający poprawnie;

S (Should have) wymagania bardzo istotne, jednak niewpływające bezpośrednio na minimalną zdolność operacyjną systemu;

C (Could have) wymagania opcjonalne, podnoszące wygodę użytkowania lub wartość biznesową rozwiązania;

W (Won't have this time) wymagania odłożone na przyszłość, które w bieżącej iteracji nie będą realizowane.

Tak przeprowadzona analiza pozwala w sposób uporządkowany opisać zarówno zakres funkcjonalny systemu, jak i ograniczenia oraz oczekiwania niefunkcjonalne, a także zewnętrzne uwarunkowania jego działania.

4.1 Przypadki użycia

4.1.1 Aktorzy

Niniejszy rozdział przedstawia aktorów wraz z opisami.

Użytkownik systemu - Reprezentuje każdą osobę korzystającą z aplikacji.

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

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

Użytkownik premium - Użytkownik z wykupioną subskrypcją: ma dostęp do funkcji premium np. oznaczenie stref **PANSA** na mapie.

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

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

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

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

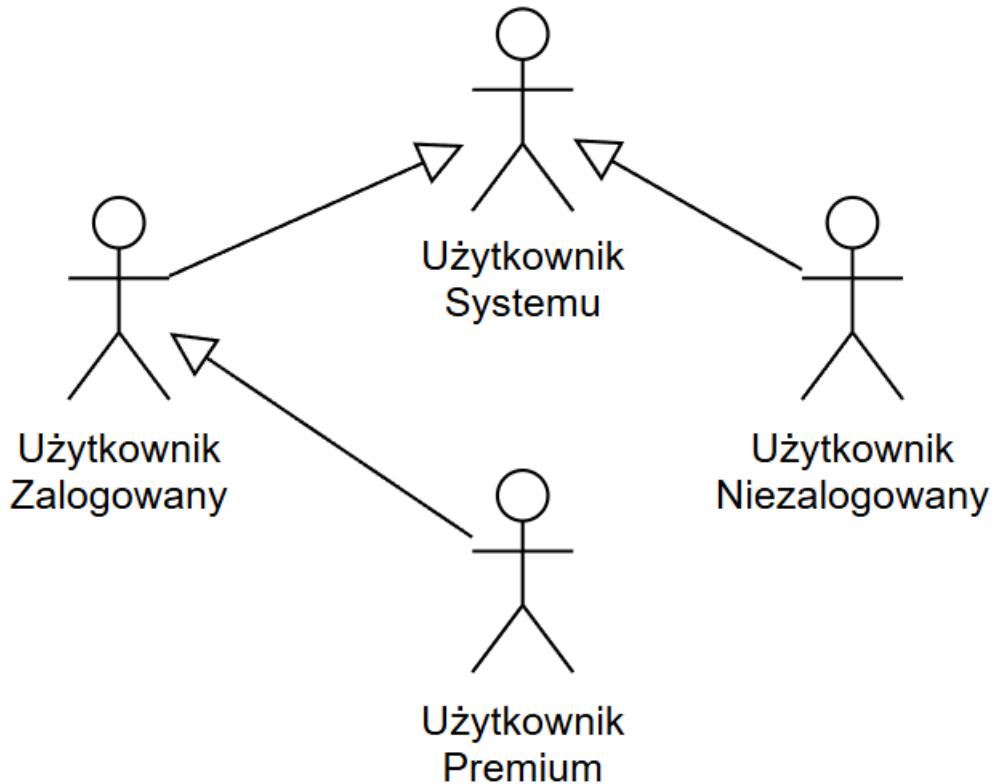
Usługa do przechowywania plików w chmurze - magazyn plików w chmurze służący do przechowywania załączników i multimedialnych użytkowników.

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

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

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

Usługa do określania strefy czasowej - usługa ustalająca strefę czasową spocinającą podstawie jego współrzędnych geograficznych.

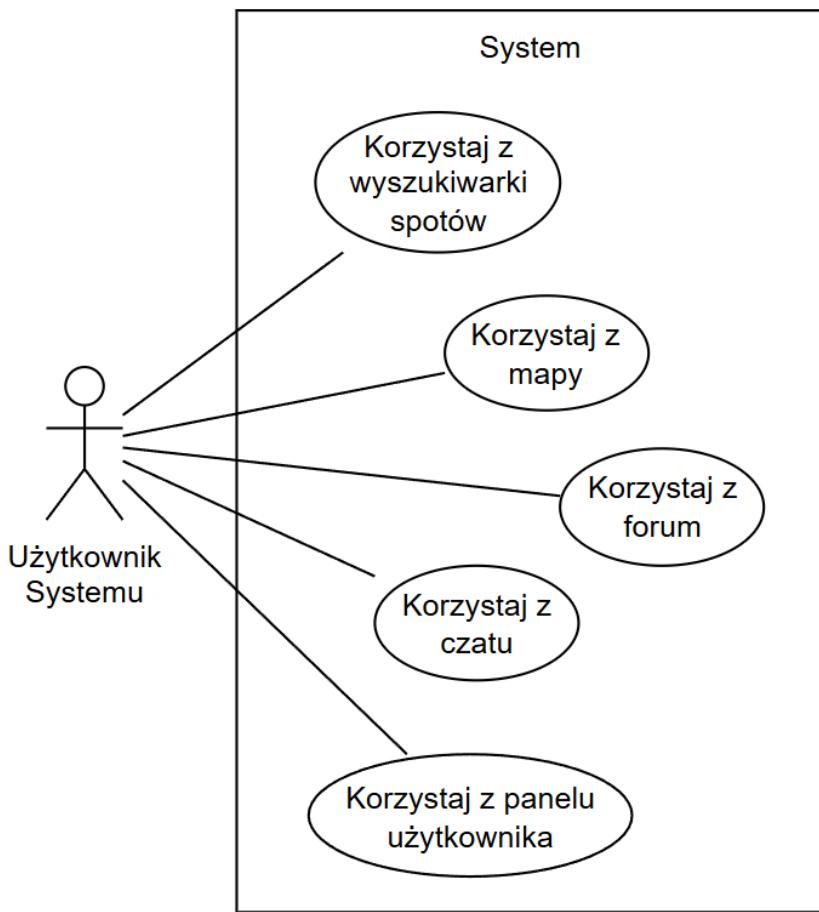


Rysunek 4.1: Diagram hierarchii użytkowników systemu

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

4.1.2 Diagramy przypadków użycia

Niniejszy rozdział przedstawia diagramy przypadków użycia.



Rysunek 4.2: Wysokopoziomowy diagram przypadków użycia

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



Rysunek 4.3: Diagram przypadków użycia dla użytkownika niezalogowanego



Rysunek 4.4: Diagram przypadków użycia dla użytkownika zalogowanego



Rysunek 4.5: Diagram przypadków użycia dla użytkownika premium



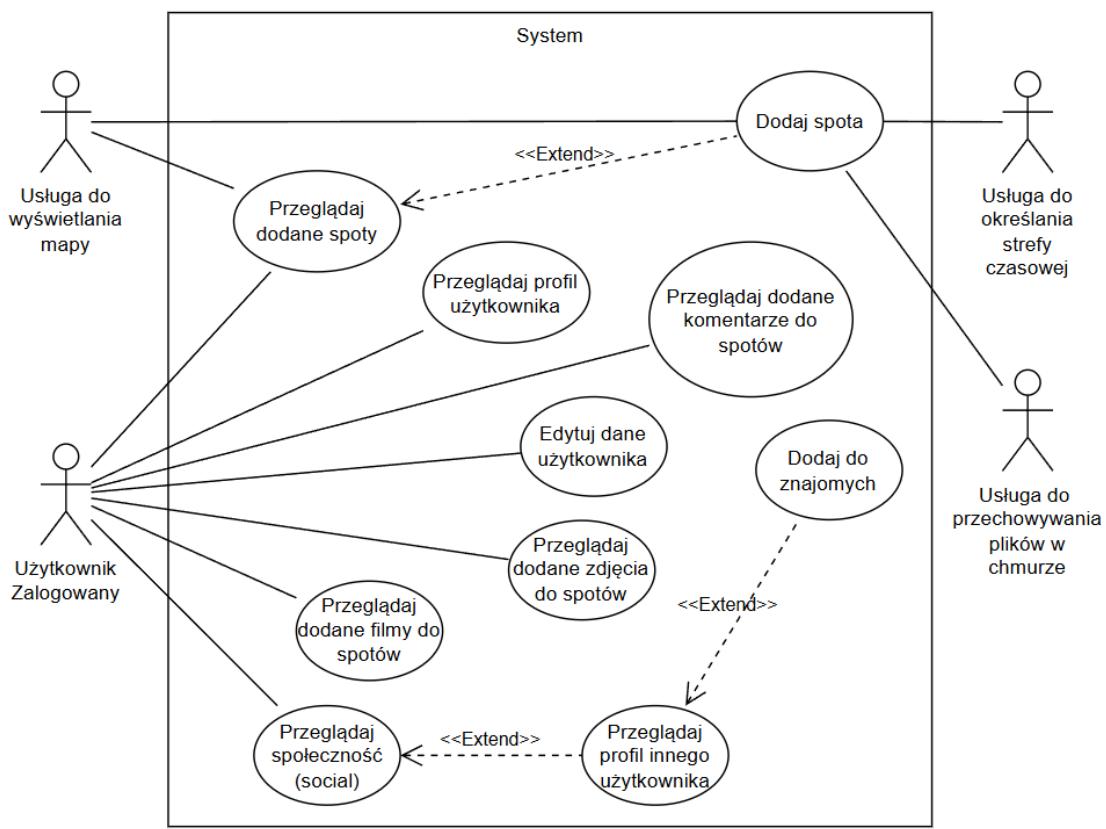
Rysunek 4.6: Diagram przypadków użycia wyszukiwarki spotów oraz mapy



Rysunek 4.7: Diagram przypadków użycia forum



Rysunek 4.8: Diagram przypadków użycia czatu



Rysunek 4.9: Diagram przypadków użycia profilu użytkownika

4.1.3 Scenariusze przypadków użycia

Scenariusze przypadków użycia – funkcje ogólne

| KARTA SCENARIUSZA PRZYPADKU UŻYCIA | |
|------------------------------------|---------------------------------------------------------|
| Identyfikator: | PU1 |
| Nazwa: | Rejestracja użytkownika |
| Priorytet: | Wysoki |
| Aktorzy: | Użytkownik niezalogowany |
| Opis: | Użytkownik zakłada konto poprzez formularz rejestracji. |

| | |
|----------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Warunki wstępne: | Użytkownik znajduje się na stronie z formularzem rejestracji. |
| Warunki końcowe: | Użytkownik posiada konto w systemie. |
| Główny przepływ zdarzeń: | <ol style="list-style-type: none"> 1. Użytkownik wypełnia formularz rejestracyjny. 2. Użytkownik naciska przycisk rejestracji. 3. System tworzy konto użytkownika. 4. System loguje użytkownika i przenosi go na ostatnio doowiedzoną podstronę. |
| Alternatywne przepływy zdarzeń: | <ol style="list-style-type: none"> 1a. Podane dane są niepoprawne – system wyświetla komunikat o błędzie oraz podświetla pola wymagające poprawy. 2a. Nazwa użytkownika jest już zajęta – system wyświetla komunikat o błędzie. 2b. Adres email jest już zajęty – system wyświetla komunikat o błędzie. |

Tabela 4.1: Scenariusz przypadku użycia: Rejestracja użytkownika

| KARTA SCENARIUSZA PRZYPADKU UŻYCIA | |
|------------------------------------|----------------------------------------------------------|
| Identyfikator: | PU2 |
| Nazwa: | Logowanie użytkownika |
| Priorytet: | Wysoki |
| Aktorzy: | Użytkownik niezalogowany |
| Opis: | Użytkownik loguje się do systemu, podając login i hasło. |
| Warunki wstępne: | Użytkownik znajduje się na stronie logowania. |

| | |
|----------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Warunki końcowe: | Użytkownik jest zalogowany i przeniesiony na ostatnio do-wiedzoną podstronę. |
| Główny przepływ zdarzeń: | <ol style="list-style-type: none"> 1. Użytkownik wypełnia formularz logowania. 2. Użytkownik naciska przycisk logowania. 3. System loguje użytkownika i przenosi go na ostatnio do-wiedzoną podstronę. |
| Alternatywne przepływy zdarzeń: | <ol style="list-style-type: none"> <li value="2">2a. Podane dane są niepoprawne – system wyświetla komunikat o błędzie. |

Tabela 4.2: Scenariusz przypadku użycia: Logowanie użytkownika

| KARTA SCENARIUSZA PRZYPADKU UŻYCIA | |
|-------------------------------------------|------------------------------------------------------------------------------|
| Identyfikator: | PU3 |
| Nazwa: | Wykupienie subskrypcji premium |
| Priorytet: | Niski |
| Aktorzy: | Użytkownik zalogowany, Bramka płatnicza, System finansowo-księgowy |
| Opis: | Użytkownik opłaca subskrypcję premium w celu uzyskania dodatkowych funkcji. |
| Warunki wstępne: | Użytkownik jest zalogowany i znajduje się w module subskrypcji. |
| Warunki końcowe: | Subskrypcja premium jest aktywna, a użytkownik ma dostęp do funkcji premium. |

| | |
|----------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Główny przepływ zdarzeń: | <ol style="list-style-type: none"> 1. Użytkownik wybiera plan subskrypcji. 2. Użytkownik przechodzi do bramki płatniczej. 3. Użytkownik podaje dane płatnicze i zatwierdza transakcję. 4. Bramka płatnicza przetwarza płatność i zwraca wynik do systemu. 5. System zapisuje informację o opłaconej subskrypcji i aktualizuje uprawnienia. 6. System generuje wpis w systemie finansowo-księgowym. |
| Alternatywne przepływy zdarzeń: | <ol style="list-style-type: none"> 4a. Płatność nie powiodła się – system informuje użytkownika i umożliwia ponowną próbę. 5a. W czasie aktualizacji subskrypcji wystąpił błąd – system cofa zmiany i wyświetla komunikat o problemie. |

Tabela 4.3: Scenariusz przypadku użycia: Wykupienie subskrypcji premium

| KARTA SCENARIUSZA PRZYPADKU UŻYCIA | |
|-------------------------------------------|----------------------------------------------------------------------------|
| Identyfikator: | PU4 |
| Nazwa: | Resetowanie hasła |
| Priorytet: | Wysoki |
| Aktorzy: | Użytkownik niezalogowany, Usługa SMTP |
| Opis: | Użytkownik inicjuje reset hasła, aby odzyskać dostęp do konta. |
| Warunki wstępne: | Użytkownik znajduje się na ekranie resetu hasła. |
| Warunki końcowe: | Użytkownik otrzymuje wiadomość e-mail z linkiem do ustalenia nowego hasła. |

| | |
|----------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Główny przepływ zdarzeń: | <ol style="list-style-type: none"> 1. Użytkownik wpisuje adres e-mail powiązany z kontem. 2. Użytkownik zatwierdza żądanie resetu hasła. 3. System generuje token resetu hasła. 4. System wysyła e-mail z linkiem do zmiany hasła. |
| Alternatywne przepływy zdarzeń: | <ol style="list-style-type: none"> 2a. Nie istnieje konto dla podanego adresu – system wyświetla komunikat o błędzie. 4a. Występuje błąd połączenia z usługą SMTP – system informuje użytkownika o problemie technicznym. |

Tabela 4.4: Scenariusz przypadku użycia: Resetowanie hasła

| KARTA SCENARIUSZA PRZYPADKU UŻYCIA | |
|-------------------------------------------|---------------------------------------------------------------------------|
| Identyfikator: | PU5 |
| Nazwa: | Zmiana hasła w ustawieniach konta |
| Priorytet: | Wysoki |
| Aktorzy: | Użytkownik zalogowany |
| Opis: | Użytkownik zmienia hasło do konta z poziomu ustawień profilu. |
| Warunki wstępne: | Użytkownik jest zalogowany i znajduje się na ekranie zmiany danych konta. |
| Warunki końcowe: | Hasło do konta użytkownika zostało zaktualizowane. |

| | |
|----------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Główny przepływ zdarzeń: | <ol style="list-style-type: none"> 1. Użytkownik wpisuje aktualne hasło. 2. Użytkownik wpisuje nowe hasło i powtarza je. 3. Użytkownik zatwierdza formularz zmiany hasła. 4. System zapisuje nowe hasło i informuje o powodzeniu operacji. |
| Alternatywne przepływy zdarzeń: | <ol style="list-style-type: none"> 3a. Aktualne hasło jest nieprawidłowe – system wyświetla komunikat i nie zapisuje zmian. 3b. Nowe hasło nie spełnia wymagań bezpieczeństwa – system informuje o błędzie i podświetla pola do poprawy. |

Tabela 4.5: Scenariusz przypadku użycia: Zmiana hasła w ustawieniach konta

| KARTA SCENARIUSZA PRZYPADKU UŻYCIA | |
|------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Identyfikator: | PU6 |
| Nazwa: | Wylogowanie użytkownika |
| Priorytet: | Wysoki |
| Aktorzy: | Użytkownik zalogowany |
| Opis: | Użytkownik wylogowuje się z aplikacji. |
| Warunki wstępne: | Użytkownik jest zalogowany. |
| Warunki końcowe: | Sesja użytkownika została zakończona, użytkownik widzi stronę główną dla niezalogowanych. |
| Główny przepływ zdarzeń: | <ol style="list-style-type: none"> 1. Użytkownik wybiera opcję wylogowania z menu. 2. System unieważnia token dostępu użytkownika. 3. System przenosi użytkownika na stronę główną aplikacji. |

| | |
|----------------------------------------|-------------------------------------------|
| Alternatywne przepływy zdarzeń: | Brak istotnych alternatywnych przepływów. |
|----------------------------------------|-------------------------------------------|

Tabela 4.6: Scenariusz przypadku użycia: Wylogowanie użytkownika

| KARTA SCENARIUSZA PRZYPADKU UŻYCIA | |
|----------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Identyfikator: | PU7 |
| Nazwa: | Przeglądanie powiadomień |
| Priorytet: | Niski |
| Aktorzy: | Użytkownik zalogowany |
| Opis: | Użytkownik przegląda listę powiadomień. |
| Warunki wstępne: | Użytkownik jest na ekranie centra powiadomień. |
| Warunki końcowe: | Powiadomienia zostały wyświetcone, a wybrane oznaczone jako przeczytane. |
| Główny przepływ zdarzeń: | <ol style="list-style-type: none"> System wyświetla powiadomienia w odwróconym porządku chronologicznym. Użytkownik otwiera wybrane powiadomienie. System oznacza powiadomienie jako przeczytane i ewentualnie przenosi użytkownika do powiązanego widoku. |
| Alternatywne przepływy zdarzeń: | <ol style="list-style-type: none"> System nie może pobrać powiadomień (błąd serwera) – użytkownik otrzymuje komunikat o błędzie i może spróbować ponownie. |

Tabela 4.7: Scenariusz przypadku użycia: Przeglądanie powiadomień

Scenariusze przypadków użycia dla funkcji premium

| KARTA SCENARIUSZA PRZYPADKU UŻYCIA | |
|----------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Identyfikator: | PU8 |
| Nazwa: | Przeszukiwanie historii czatu |
| Priorytet: | Niski |
| Aktorzy: | Użytkownik premium |
| Opis: | Użytkownik wyszukuje konkretne wiadomości w historii czatu. |
| Warunki wstępne: | Użytkownik jest zalogowany jako użytkownik premium i znajduje się w widoku czatu. |
| Warunki końcowe: | Wiadomości spełniające kryteria wyszukiwania zostały wyświetlane. |
| Główny przepływ zdarzeń: | <ol style="list-style-type: none"> Użytkownik otwiera pole wyszukiwania historii w czacie. Użytkownik wpisuje frazę lub filtr (np. zakres dat, autor). System filtryuje wiadomości zgodnie z kryteriami. System prezentuje listę dopasowanych fragmentów rozmowy. |
| Alternatywne przepływy zdarzeń: | 4a. Nie znaleziono wiadomości spełniających kryteria – system informuje o braku wyników wyszukwiania. |

Tabela 4.8: Scenariusz przypadku użycia: Przeszukiwanie historii czatu

| KARTA SCENARIUSZA PRZYPADKU UŻYCIA | |
|------------------------------------|-----------------------------------------|
| Identyfikator: | PU9 |
| Nazwa: | Przeglądanie wysłanych plików na czacie |

| | |
|----------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Priorytet: | Niski |
| Aktorzy: | Użytkownik premium, Usługa do przechowywania plików w chmurze |
| Opis: | Użytkownik przegląda listę plików wysłanych w ramach czatów. |
| Warunki wstępne: | Użytkownik jest zalogowany jako użytkownik premium. |
| Warunki końcowe: | Użytkownik widzi listę wysłanych plików i może przechodzić do powiązanych czatów. |
| Główny przepływ zdarzeń: | <ol style="list-style-type: none"> 1. Użytkownik otwiera sekcję „Wysłane pliki”. 2. System pobiera metadane plików z usługi przechowywania. 3. System wyświetla listę plików z podstawowymi informacjami (nazwa, typ, data). 4. Użytkownik wybiera plik, aby otworzyć go lub przejść do powiązanego czatu. |
| Alternatywne przepływy zdarzeń: | <p>4a. Doszło do błędu podczas pobierania pliku - system wyświetla odpowiedni komunikat.</p> |

Tabela 4.9: Scenariusz przypadku użycia: Przeglądanie wysłanych plików na czacie

| KARTA SCENARIUSZA PRZYPADKU UŻYCIA | |
|------------------------------------|------------------|
| Identyfikator: | PU10 |
| Nazwa: | Zmiana typu mapy |
| Priorytet: | Niski |

| | |
|----------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Aktorzy: | Użytkownik premium, Usługa do wyświetlania mapy |
| Opis: | Użytkownik zmienia typ mapy (np. standardowa, satelitarna, hybrydowa). |
| Warunki wstępne: | Użytkownik premium jest na ekranie mapy. |
| Warunki końcowe: | Mapa jest wyświetlana w wybranym typie. |
| Główny przepływ zdarzeń: | <ol style="list-style-type: none"> 1. Użytkownik otwiera ustawienia widoku mapy. 2. Użytkownik wybiera typ mapy z dostępnej listy. 3. System przełącza widok mapy na wybrany typ. |
| Alternatywne przepływy zdarzeń: | <p>3a. Wybrany typ mapy nie jest dostępny (błąd usługi mapowej) – system przywraca poprzedni typ i informuje o błędzie.</p> |

Tabela 4.10: Scenariusz przypadku użycia: Zmiana typu mapy

| KARTA SCENARIUSZA PRZYPADKU UŻYCIA | |
|------------------------------------|---------------------------------------------------------------------|
| Identyfikator: | PU11 |
| Nazwa: | Przeglądanie stref PANSA |
| Priorytet: | Niski |
| Aktorzy: | Użytkownik premium, Usługa do wyświetlania mapy |
| Opis: | Użytkownik wyświetla na mapie strefy przestrzeni powietrznej PANSA. |
| Warunki wstępne: | Użytkownik premium ma otwarty moduł mapy. |

| | |
|----------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Warunki końcowe: | Strefy PANSA zostały zwizualizowane na mapie. |
| Główny przepływ zdarzeń: | <ol style="list-style-type: none"> 1. Użytkownik włącza warstwę „Strefy PANSA”. 2. System pobiera dane o strefach. 3. System nakłada kontury stref na mapę. |
| Alternatywne przepływy zdarzeń: | <ol style="list-style-type: none"> <li value="2">2a. Dane o strefach są chwilowo niedostępne – system komunikuje problem i nie włącza warstwy. |

Tabela 4.11: Scenariusz przypadku użycia: Przeglądanie stref PANSA

Scenariusze przypadków użycia dla wyszukiwarki

| KARTA SCENARIUSZA PRZYPADKU UŻYCIA | |
|-------------------------------------------|--------------------------------------------------------------------------|
| Identyfikator: | PU12 |
| Nazwa: | Wyszukiwanie spota w globalnej wyszukiwarce |
| Priorytet: | Wysoki |
| Aktorzy: | Użytkownik niezalogowany, Usługa do wyświetlania mapy, Usługa do pogody |
| Opis: | Użytkownik wyszukuje spoty za pomocą globalnej wyszukiwarki w aplikacji. |
| Warunki wstępne: | Użytkownik znajduje się na stronie głównej z wyszukiwarką. |
| Warunki końcowe: | Użytkownik otrzymuje listę znalezionych spotów. |

| | |
|----------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| Główny przepływ zdarzeń: | 1. Użytkownik wpisuje frazę w globalnej wyszukiwarce. 2. System wyszukuje spedy spełniające kryteria. 3. System wyświetla listę wyników. |
| Alternatywne przepływy zdarzeń: | 3a. System informuje o braku wyników spełniających kryteria. |

Tabela 4.12: Scenariusz przypadku użycia: Wyszukiwanie spota w globalnej wyszukiwarce

| KARTA SCENARIUSZA PRZYPADKU UŻYCIA | |
|------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Identyfikator: | PU13 |
| Nazwa: | Przejście do spota na mapie z wyszukiwarki |
| Priorytet: | Wysoki |
| Aktorzy: | Użytkownik niezalogowany |
| Opis: | Użytkownik przechodzi z wyników wyszukiwarki do widoku mapy ustawionego na konkretny spot. |
| Warunki wstępne: | Wyświetlona jest lista wyników wyszukiwania spotów. |
| Warunki końcowe: | Mapa jest przybliżona do wybranego spota, a jego szczegóły są dostępne. |
| Główny przepływ zdarzeń: | 1. Użytkownik wybiera spota z listy wyników. 2. System przełącza widok na moduł mapy. 3. System ustawia mapę na lokalizację spota i otwiera jego szczegóły. |

| | |
|----------------------------------------|-------------------------------------------|
| Alternatywne przepływy zdarzeń: | Brak istotnych alternatywnych przepływów. |
|----------------------------------------|-------------------------------------------|

Tabela 4.13: Scenariusz przypadku użycia: Przejście do spota na mapie z wyszukiwarki

Scenariusze przypadków użycia dla mapy

| KARTA SCENARIUSZA PRZYPADKU UŻYCIA | |
|----------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Identyfikator: | PU14 |
| Nazwa: | Przeglądanie mapy spotów |
| Priorytet: | Wysoki |
| Aktorzy: | Użytkownik niezalogowany, Usługa do wyświetlania mapy |
| Opis: | Użytkownik przegląda mapę spotów. |
| Warunki wstępne: | Użytkownik znajduje się w module mapy. |
| Warunki końcowe: | Mapa ze spotami została wyświetlona, a użytkownik może przybliżać, oddalać i przesuwać widok. |
| Główny przepływ zdarzeń: | <ol style="list-style-type: none"> System inicjuje widok mapy z domyślnym obszarem. System pobiera listę spotów. System rysuje znaczniki spotów na mapie. |
| Alternatywne przepływy zdarzeń: | <ol style="list-style-type: none"> Usługa mapy jest niedostępna – system wyświetla komunikat o błędzie. |

Tabela 4.14: Scenariusz przypadku użycia: Przeglądanie mapy spotów

| KARTA SCENARIUSZA PRZYPADKU UŻYCIA | |
|----------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Identyfikator: | PU15 |
| Nazwa: | Otwarcie szczegółów spota |
| Priorytet: | Wysoki |
| Aktorzy: | Użytkownik niezalogowany, Użytkownik zalogowany |
| Opis: | Użytkownik otwiera widok szczegółów wybranego spota. |
| Warunki wstępne: | Użytkownik widzi mapę spotów. |
| Warunki końcowe: | Wyświetlony został widok szczegółów spota z podstawowymi informacjami oraz jego lokalizacją na mapie. |
| Główny przepływ zdarzeń: | <ol style="list-style-type: none"> 1. Użytkownik wybiera spota z mapy. 2. System pobiera dane szczegółowe spota (informacje opisowe, lokalizacja). 3. System otwiera widok szczegółów spota. |
| Alternatywne przepływy zdarzeń: | <ol style="list-style-type: none"> 2a. Spot nie istnieje (został usunięty lub ukryty) – system informuje użytkownika i powraca do poprzedniego widoku. 2b. Wystąpił błąd podczas pobierania danych spota – system wyświetla komunikat o błędzie i umożliwia ponowną próbę. |

Tabela 4.15: Scenariusz przypadku użycia: Otwarcie szczegółów spota

| KARTA SCENARIUSZA PRZYPADKU UŻYCIA | |
|------------------------------------|----------------------------------|
| Identyfikator: | PU16 |
| Nazwa: | Przeglądanie komentarzy do spota |
| Priorytet: | Średni |
| Aktorzy: | Użytkownik niezalogowany |

| | |
|----------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Opis: | Użytkownik czyta komentarze pod wybranym spotem. |
| Warunki wstępne: | Wyświetlany jest widok szczegółów spota. |
| Warunki końcowe: | Lista komentarzy do spota została wyświetlona. |
| Główny przepływ zdarzeń: | <ol style="list-style-type: none"> 1. System pobiera komentarze powiązane ze spotem. 2. System wyświetla komentarze w kolejności chronologicznej lub według popularności. 3. Użytkownik przewija listę komentarzy. |
| Alternatywne przepływy zdarzeń: | 1a. Spot nie ma jeszcze komentarzy – system wyświetla odpowiednią informację. |

Tabela 4.16: Scenariusz przypadku użycia: Przeglądanie komentarzy do spota

| KARTA SCENARIUSZA PRZYPADKU UŻYCIA | |
|------------------------------------|------------------------------------------------------------|
| Identyfikator: | PU17 |
| Nazwa: | Przeglądanie pogody na spocie |
| Priorytet: | Średni |
| Aktorzy: | Użytkownik niezalogowany, Usługa danych pogodowych |
| Opis: | Użytkownik sprawdza prognozę pogody dla lokalizacji spota. |
| Warunki wstępne: | Wyświetlany jest widok szczegółów spota. |
| Warunki końcowe: | Prognoza pogody dla spota została wyświetlona. |

| | |
|----------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Główny przepływ zdarzeń: | <ol style="list-style-type: none"> 1. Użytkownik otwiera zakładkę pogody. 2. System wysyła zapytanie do usługi pogodowej z lokalizacją spota. 3. System odbiera prognozę i prezentuje ją (temperatura, prędkość wiatru, opady). |
| Alternatywne przepływy zdarzeń: | <p>2a. Usługa pogodowa jest niedostępna – system wyświetla komunikat o braku danych pogodowych.</p> |

Tabela 4.17: Scenariusz przypadku użycia: Przeglądanie pogody na spocie

| KARTA SCENARIUSZA PRZYPADKU UŻYCIA | |
|------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Identyfikator: | PU18 |
| Nazwa: | Wyszukiwanie spota na mapie |
| Priorytet: | Wysoki |
| Aktorzy: | Użytkownik niezalogowany |
| Opis: | Użytkownik wyszukuje spota po nazwie korzystając z pola wyszukiwania na mapie. |
| Warunki wstępne: | Użytkownik widzi mapę spotów. |
| Warunki końcowe: | Mapa zostaje ustawiona na wybranego spota lub listę dopasowań. |
| Główny przepływ zdarzeń: | <ol style="list-style-type: none"> 1. Użytkownik wpisuje frazę w polu wyszukiwania na mapie. 2. System podpowiada listę pasujących spotów. 3. Użytkownik wybiera spota z listy. 4. System przenosi użytkownika na mapie do wybranego spota. |

| | |
|----------------------------------------|--------------------------------------------------------------------------------------|
| Alternatywne przepływy zdarzeń: | 2a. Brak wyników dla podanej frazy – system informuje użytkownika o braku dopasowań. |
|----------------------------------------|--------------------------------------------------------------------------------------|

Tabela 4.18: Scenariusz przypadku użycia: Wyszukiwanie spota na mapie

Scenariusze przypadków użycia dla czatu

| KARTA SCENARIUSZA PRZYPADKU UŻYCIA | |
|----------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Identyfikator: | PU19 |
| Nazwa: | Utworzenie prywatnego czatu |
| Priorytet: | Wysoki |
| Aktorzy: | Użytkownik zalogowany |
| Opis: | Użytkownik tworzy prywatną konwersację z innym użytkownikiem. |
| Warunki wstępne: | Użytkownik jest zalogowany i znajduje się w zakładce społeczność. |
| Warunki końcowe: | Nowy czat prywatny został utworzony i wyświetlony użytkownikowi. |
| Główny przepływ zdarzeń: | <ol style="list-style-type: none"> Użytkownik wybiera opcję utworzenia nowego czatu. System tworzy nowy czat (jeśli nie istnieje). System otwiera widok nowego czatu. |
| Alternatywne przepływy zdarzeń: | <ol style="list-style-type: none"> a. Taki czat już istnieje – system zamiast tworzyć nowy, otwiera istniejącą konwersację. |

Tabela 4.19: Scenariusz przypadku użycia: Utworzenie prywatnego czatu

| KARTA SCENARIUSZA PRZYPADKU UŻYCIA | |
|----------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Identyfikator: | PU20 |
| Nazwa: | Otworzenie czatu |
| Priorytet: | Wysoki |
| Aktorzy: | Użytkownik zalogowany |
| Opis: | Użytkownik otwiera wybrany czat, aby wyświetlić historię rozmowy i móc wysyłać kolejne wiadomości. |
| Warunki wstępne: | Użytkownik jest zalogowany i widzi listę swoich czatów lub otrzymał powiadomienie prowadzące do czatu. |
| Warunki końcowe: | Wybrany czat został otworzony, a historia rozmowy jest wiadoma dla użytkownika. |
| Główny przepływ zdarzeń: | <ol style="list-style-type: none"> Użytkownik wybiera czat z listy czatów lub z powiadomienia. System pobiera dane czatu (uczestników, ostatnie wiadomości). System oznacza nieprzeczytane wiadomości na czacie jako przeczytane. System wyświetla widok czatu wraz z historią rozmowy. |
| Alternatywne przepływy zdarzeń: | <p>2a. Czat nie jest już dostępny (np. został usunięty lub użytkownik utracił do niego dostęp) – system wyświetla komunikat o braku dostępu i powraca do listy czatów.</p> <p>2b. Wystąpił błąd podczas pobierania danych czatu – system wyświetla komunikat o błędzie i umożliwia ponowną próbę.</p> |

Tabela 4.20: Scenariusz przypadku użycia: Otworzenie czatu

| KARTA SCENARIUSZA PRZYPADKU UŻYCIA | |
|----------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Identyfikator: | PU21 |
| Nazwa: | Utworzenie czatu grupowego |
| Priorytet: | Średni |
| Aktorzy: | Użytkownik zalogowany |
| Opis: | Użytkownik tworzy nowy czat grupowy z kilkoma uczestnikami. |
| Warunki wstępne: | Użytkownik jest zalogowany i znajduje się na dowolnym czasie prywatnym. |
| Warunki końcowe: | Czat grupowy został utworzony i wyświetlony na ekranie. |
| Główny przepływ zdarzeń: | <ol style="list-style-type: none"> 1. Użytkownik wybiera opcję utworzenia czatu grupowego. 2. Użytkownik wybiera uczestników grupy. 3. Użytkownik zatwierdza utworzenie czatu. 4. System tworzy czat grupowy i dodaje do niego wskazanych użytkowników. 5. System otwiera widok nowego czatu grupowego. |
| Alternatywne przepływy zdarzeń: | <p>3a. System nie może utworzyć czatu – aplikacja informuje o błędzie.</p> |

Tabela 4.21: Scenariusz przypadku użycia: Utworzenie czatu grupowego

| KARTA SCENARIUSZA PRZYPADKU UŻYCIA | |
|------------------------------------|---------------------------|
| Identyfikator: | PU22 |
| Nazwa: | Przeglądanie listy czatów |
| Priorytet: | Wysoki |

| | |
|----------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Aktorzy: | Użytkownik zalogowany |
| Opis: | Użytkownik przegląda listę swoich czatów prywatnych i grupowych. |
| Warunki wstępne: | Użytkownik jest zalogowany i otwiera moduł czatu. |
| Warunki końcowe: | Lista czatów użytkownika została wyświetlona. |
| Główny przepływ zdarzeń: | <ol style="list-style-type: none"> 1. System pobiera listę czatów użytkownika. 2. System wyświetla listę czatów z podstawowymi informacjami. 3. Użytkownik wybiera czat z listy. 4. System otwiera widok wybranego czatu. |
| Alternatywne przepływy zdarzeń: | Brak istotnych alternatywnych przepływów. |

Tabela 4.22: Scenariusz przypadku użycia: Przeglądanie listy czatów

| KARTA SCENARIUSZA PRZYPADKU UŻYCIA | |
|------------------------------------|-----------------------------------------------------------------------|
| Identyfikator: | PU23 |
| Nazwa: | Wysyłanie wiadomości na czacie |
| Priorytet: | Wysoki |
| Aktorzy: | Użytkownik zalogowany |
| Opis: | Użytkownik wysyła wiadomość tekstową na czacie. |
| Warunki wstępne: | Użytkownik jest zalogowany i znajduje się w widoku konkretnego czatu. |

| | |
|----------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Warunki końcowe: | Nowa wiadomość jest zapisana i widoczna w historii czatu. |
| Główny przepływ zdarzeń: | <ol style="list-style-type: none"> 1. Użytkownik wpisuje treść wiadomości. 2. Użytkownik wysyła wiadomość. 3. System zapisuje wiadomość i dostarcza ją do uczestników czatu. 4. System wyświetla wiadomość na liście wiadomości. |
| Alternatywne przepływy zdarzeń: | <p>2a. Treść wiadomości jest pusta – system blokuje wysłanie i pozostaje w tym samym widoku.</p> |

Tabela 4.23: Scenariusz przypadku użycia: Wysyłanie wiadomości na czacie

| KARTA SCENARIUSZA PRZYPADKU UŻYCIA | |
|------------------------------------|-----------------------------------------------------------|
| Identyfikator: | PU24 |
| Nazwa: | Wysyłanie GIF-a na czacie |
| Priorytet: | Średni |
| Aktorzy: | Użytkownik zalogowany, Usługa GIF-ów |
| Opis: | Użytkownik wysyła animację GIF w konwersacji czatowej. |
| Warunki wstępne: | Użytkownik jest zalogowany i znajduje się w widoku czatu. |
| Warunki końcowe: | Wybrany GIF został dodany jako wiadomość w czacie. |

| | |
|----------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Główny przepływ zdarzeń: | <ol style="list-style-type: none"> 1. Użytkownik wybiera opcję dodania GIF-a. 2. System otwiera okno wyszukiwarki GIF-ów. 3. Użytkownik wybiera lub wyszukuje GIF-a. 4. Użytkownik zatwierdza wysłanie GIF-a. 5. System dodaje GIF-a jako wiadomość na czacie. |
| Alternatywne przepływy zdarzeń: | <p>2a. Usługa GIF-ów jest niedostępna – system informuje o braku możliwości wysłania GIF-a.</p> |

Tabela 4.24: Scenariusz przypadku użycia: Wysyłanie GIF-a na czacie

| KARTA SCENARIUSZA PRZYPADKU UŻYCIA | |
|------------------------------------|---------------------------------------------------------------------|
| Identyfikator: | PU25 |
| Nazwa: | Wysyłanie pliku na czacie |
| Priorytet: | Średni |
| Aktorzy: | Użytkownik zalogowany, Usługa do przechowywania plików w chmurze |
| Opis: | Użytkownik wysyła plik (np. zdjęcie, film) na czacie. |
| Warunki wstępne: | Użytkownik jest zalogowany i znajduje się w widoku czatu. |
| Warunki końcowe: | Plik został zapisany w chmurze i powiązany z wiadomością na czacie. |

| | |
|----------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Główny przepływ zdarzeń: | <ol style="list-style-type: none"> 1. Użytkownik wybiera opcję dodania pliku. 2. Użytkownik wybiera plik z urządzenia. 3. System przesyła plik do usługi przechowywania w chmurze. 4. System tworzy wiadomość z odnośnikiem do pliku. 5. System wyświetla wiadomość na liście czatu. |
| Alternatywne przepływy zdarzeń: | <p>3a. Przesyłanie pliku nie powiodło się – system informuje użytkownika i umożliwia ponowną próbę.</p> |

Tabela 4.25: Scenariusz przypadku użycia: Wysyłanie pliku na czacie

| KARTA SCENARIUSZA PRZYPADKU UŻYCIA | |
|-------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Identyfikator: | PU26 |
| Nazwa: | Edycja ustawień czatu |
| Priorytet: | Niski |
| Aktorzy: | Użytkownik zalogowany |
| Opis: | Użytkownik modyfikuje ustawienia czatu (np. nazwę, avatar, tryb powiadomień). |
| Warunki wstępne: | Użytkownik jest zalogowany i ma uprawnienia do edycji danego czatu. |
| Warunki końcowe: | Zaktualizowane ustawienia czatu są zapisane i widoczne dla uczestników. |
| Główny przepływ zdarzeń: | <ol style="list-style-type: none"> 1. Użytkownik otwiera panel ustawień czatu. 2. Użytkownik wprowadza zmiany (np. nazwę, opis, avatar). 3. Użytkownik zapisuje zmiany. 4. System waliduje dane i aktualizuje konfigurację czatu. |

| | |
|----------------------------------------|--------------------------------------------------------------|
| Alternatywne przepływy zdarzeń: | Brak istotnych alternatywnych przepływów poza walidacja pól. |
|----------------------------------------|--------------------------------------------------------------|

Tabela 4.26: Scenariusz przypadku użycia: Edycja ustawień czatu

| KARTA SCENARIUSZA PRZYPADKU UŻYCIA | |
|----------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Identyfikator: | PU27 |
| Nazwa: | Dodanie członka do czatu grupowego |
| Priorytet: | Średni |
| Aktorzy: | Użytkownik zalogowany |
| Opis: | Użytkownik dodaje nowego uczestnika do czatu grupowego. |
| Warunki wstępne: | Użytkownik jest zalogowany i znajduje się w czacie grupowym. |
| Warunki końcowe: | Nowy uczestnik został dodany do czatu grupowego. |
| Główny przepływ zdarzeń: | <ol style="list-style-type: none"> Użytkownik otwiera listę uczestników czatu grupowego. Użytkownik wybiera opcję dodania nowego członka. Użytkownik wskazuje użytkownika do dodania i zatwierdza wybór. System dodaje wskazanego użytkownika do czatu grupowego. |
| Alternatywne przepływy zdarzeń: | <ol style="list-style-type: none"> Operacja nie powiodła się – system informuje o błędzie. |

Tabela 4.27: Scenariusz przypadku użycia: Dodanie członka do czatu grupowego

Scenariusze przypadków użycia dla forum

| KARTA SCENARIUSZA PRZYPADKU UŻYCIA | |
|----------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Identyfikator: | PU28 |
| Nazwa: | Przeglądanie postów na forum |
| Priorytet: | Wysoki |
| Aktorzy: | Użytkownik niezalogowany, Użytkownik zalogowany |
| Opis: | Użytkownik przegląda listę postów na forum z możliwością sortowania wyników. |
| Warunki wstępne: | Użytkownik znajduje się w module forum. |
| Warunki końcowe: | Lista postów forum jest wyświetlona, a użytkownik może przechodzić do szczegółów wybranego posta. |
| Główny przepływ zdarzeń: | <ol style="list-style-type: none"> Użytkownik przechodzi do widoku listy postów forum. System pobiera listę postów i domyślnie wyświetla je w kolejności od najnowszych. Użytkownik wybiera sposób sortowania listy. System aktualizuje listę postów zgodnie z wybranym kryterium sortowania. Użytkownik wybiera post, który chce przeczytać. System otwiera szczegółowy widok wybranego posta. |
| Alternatywne przepływy zdarzeń: | <p>6a. System nie może pobrać szczegółów posta – system wyświetla komunikat o błędzie.</p> |

Tabela 4.28: Scenariusz przypadku użycia: Przeglądanie postów na forum

KARTA SCENARIUSZA PRZYPADKU UŻYCIA

| | |
|----------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Identyfikator: | PU29 |
| Nazwa: | Wyszukiwanie postów na forum |
| Priorytet: | Średni |
| Aktorzy: | Użytkownik niezalogowany, Użytkownik zalogowany |
| Opis: | Użytkownik wyszukuje posty na forum na podstawie tytułu, kategorii, tagów oraz autora. |
| Warunki wstępne: | Użytkownik znajduje się w module forum lub na stronie wyszukiwarki postów. |
| Warunki końcowe: | Lista postów spełniających zadane kryteria wyszukiwania jest wyświetlona. |
| Główny przepływ zdarzeń: | <ol style="list-style-type: none"> Użytkownik otwiera panel wyszukiwania postów na forum. Użytkownik określa kryteria wyszukiwania (np. tytuł, kategoria, tagi, autor). Użytkownik uruchamia wyszukiwanie. System filtryuje posty zgodnie z podanymi kryteriami i wyświetla listę wyników. |
| Alternatywne przepływy zdarzeń: | <p>4a. Brak postów spełniających zadane kryteria – system wyświetla informację o braku wyników.</p> <p>4b. Wystąpił błąd podczas wyszukiwania – system wyświetla komunikat o błędzie i umożliwia ponowną próbę.</p> |

Tabela 4.29: Scenariusz przypadku użycia: Wyszukiwanie postów na forum

| KARTA SCENARIUSZA PRZYPADKU UŻYCIA | |
|------------------------------------|------------------------|
| Identyfikator: | PU30 |
| Nazwa: | Dodanie posta na forum |

| | |
|----------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Priorytet: | Wysoki |
| Aktorzy: | Użytkownik zalogowany, Usługa do przechowywania plików w chmurze |
| Opis: | Użytkownik publikuje nowy post na forum, określając jego treść, kategorię, tagi oraz opcjonalne załączniki. |
| Warunki wstępne: | Użytkownik znajduje się w module forum. |
| Warunki końcowe: | Nowy post jest poprawnie zapisany i widoczny na forum. |
| Główny przepływ zdarzeń: | <ol style="list-style-type: none"> 1. Użytkownik wybiera opcję dodania nowego posta. 2. Użytkownik wpisuje tytuł i treść posta. 3. Użytkownik wybiera kategorię posta. 4. (Opcjonalnie) Użytkownik wybiera tagi przypisane do posta. 5. (Opcjonalnie) Użytkownik dodaje załączniki (zdjęcia / filmy) do posta. 6. Użytkownik publikuje posta. 7. System zapisuje posta (oraz poprawne załączniki w chmurze) i wyświetla go na liście postów. |
| Alternatywne przepływy zdarzeń: | <p>6a. Załącznik nie może zostać zapisany lub nie spełnia wymagań (np. zbyt duży rozmiar, nieobsługiwany format) – system informuje użytkownika o błędzie, blokuje publikację posta i wymaga usunięcia lub podmiany problematycznego pliku.</p> <p>6b. Formularz zawiera błędne lub niekompletne dane (np. brak tytułu lub treści) – system wyświetla komunikat i prosi o poprawę danych przed publikacją.</p> |

Tabela 4.30: Scenariusz przypadku użycia: Dodanie posta na forum

| KARTA SCENARIUSZA PRZYPADKU UŻYCIA | |
|----------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Identyfikator: | PU31 |
| Nazwa: | Dodanie komentarza na forum |
| Priorytet: | Wysoki |
| Aktorzy: | Użytkownik zalogowany |
| Opis: | Użytkownik dodaje komentarz pod postem na forum, opcjonalnie z załącznikami (zdjęcia/filmy). |
| Warunki wstępne: | Użytkownik jest zalogowany i widzi szczegóły posta. |
| Warunki końcowe: | Nowy komentarz został zapisany i jest widoczny pod postem. |
| Główny przepływ zdarzeń: | <ol style="list-style-type: none"> Użytkownik wpisuje treść komentarza w formularzu pod postem. (Opcjonalnie) Użytkownik dodaje załączniki (zdjęcia/filmy) do komentarza. Użytkownik publikuje komentarz. System zapisuje komentarz (oraz poprawne załączniki) i odświeża listę komentarzy. |
| Alternatywne przepływy zdarzeń: | <p>3a. Treść komentarza lub załączniki są niepoprawne (np. naruszają walidację) – system wyświetla komunikat o błędzie i blokuje publikację do czasu poprawy danych.</p> |

Tabela 4.31: Scenariusz przypadku użycia: Dodanie komentarza na forum

| KARTA SCENARIUSZA PRZYPADKU UŻYCIA | |
|------------------------------------|--------------------------------------------|
| Identyfikator: | PU32 |
| Nazwa: | Przeglądanie historii interakcji z postami |

| | |
|----------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Priorytet: | Średni |
| Aktorzy: | Użytkownik zalogowany |
| Opis: | Użytkownik przegląda historię swoich aktywności na forum (dodane posty, komentarze, reakcje). |
| Warunki wstępne: | Użytkownik jest zalogowany. |
| Warunki końcowe: | Lista interakcji użytkownika z postami jest wyświetlona. |
| Główny przepływ zdarzeń: | <ol style="list-style-type: none"> Użytkownik przechodzi do sekcji historii aktywności. System pobiera historię interakcji użytkownika. System wyświetla listę interakcji z możliwością filtrowania. |
| Alternatywne przepływy zdarzeń: | Brak istotnych alternatywnych przepływów. |

Tabela 4.32: Scenariusz przypadku użycia: Przeglądanie historii interakcji z postami

| KARTA SCENARIUSZA PRZYPADKU UŻYCIA | |
|------------------------------------|--------------------------------------------------------------------------------------------------------------------|
| Identyfikator: | PU33 |
| Nazwa: | Zarządzanie komentarzami na forum |
| Priorytet: | Niski |
| Aktorzy: | Użytkownik zalogowany |
| Opis: | Użytkownik zarządza komentarzami pod postami forum (edykcja, usuwanie, zgłaszanie komentarzy innych użytkowników). |

| | |
|----------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Warunki wstępne: | Użytkownik jest zalogowany i ma dostęp do danego wątku forum. |
| Warunki końcowe: | Komentarze zostały zaktualizowane zgodnie z działaniami użytkownika. |
| Główny przepływ zdarzeń: | <ol style="list-style-type: none"> 1. Użytkownik otwiera widok komentarzy pod postem. 2. Użytkownik wybiera komentarz i odpowiednią akcję (edytacja, usunięcie, zgłoszenie). 3. System weryfikuje uprawnienia użytkownika oraz zgodność akcji z jego rolą. 4. System wykonuje wybraną akcję (np. zapisuje zmiany, usuwa komentarz lub przygotowuje zgłoszenie) i aktualizuje widok. |
| Alternatywne przepływy zdarzeń: | <ol style="list-style-type: none"> 2a. Użytkownik próbuje zgłosić własny komentarz – system blokuje operację i informuje, że nie można zgłaszać własnych treści. 3a. Użytkownik nie ma wymaganych uprawnień do wykonania wybranej akcji – system blokuje operację i informuje o braku uprawnień. |

Tabela 4.33: Scenariusz przypadku użycia: Zarządzanie komentarzami na forum

| KARTA SCENARIUSZA PRZYPADKU UŻYCIA | |
|------------------------------------|-----------------------------------------------|
| Identyfikator: | PU34 |
| Nazwa: | Zgłoszenie komentarza naruszającego regulamin |
| Priorytet: | Średni |
| Aktorzy: | Użytkownik zalogowany |

| | |
|----------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Opis: | Użytkownik zgłasza komentarz na forum jako naruszający regulamin. |
| Warunki wstępne: | Użytkownik widzi komentarz w aplikacji. |
| Warunki końcowe: | Zgłoszenie komentarza zostało zapisane i trafiło do kolejki moderacyjnej. |
| Główny przepływ zdarzeń: | <ol style="list-style-type: none"> 1. Użytkownik wybiera opcję „Zgłoś komentarz”. 2. Użytkownik wybiera kategorię naruszenia, podaje szczegóły i potwierdza zgłoszenie. 3. System zapisuje zgłoszenie i wiąże je z komentarzem oraz zgłaszającym użytkownikiem. |
| Alternatywne przepływy zdarzeń: | <ol style="list-style-type: none"> 1a. Użytkownik próbuje zgłosić własny komentarz – system blokuje operację i wyświetla komunikat, że nie można zgłaszać własnych treści. 3a. Komentarz został już wcześniej zgłoszony – system informuje użytkownika, że komentarz znajduje się już w kolejce moderacyjnej i nie zapisuje kolejnego zgłoszenia. |

Tabela 4.34: Scenariusz przypadku użycia: Zgłoszenie komentarza naruszającego regulamin

| KARTA SCENARIUSZA PRZYPADKU UŻYCIA | |
|------------------------------------|---------------------------|
| Identyfikator: | PU35 |
| Nazwa: | Zgłoszenie posta na forum |
| Priorytet: | Średni |
| Aktorzy: | Użytkownik zalogowany |

| | |
|----------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Opis: | Użytkownik zgłasza post forum jako naruszający regulamin lub tematykę. |
| Warunki wstępne: | Wyświetlony jest widok posta na forum. |
| Warunki końcowe: | Zgłoszenie posta zostało zapisane |
| Główny przepływ zdarzeń: | <ol style="list-style-type: none"> 1. Użytkownik wybiera opcję „Zgłoś post”. 2. Użytkownik wybiera kategorię naruszenia, podaje szczegóły i potwierdza zgłoszenie. 3. System zapisuje zgłoszenie i oznacza post jako zgłoszony. |
| Alternatywne przepływy zdarzeń: | <ol style="list-style-type: none"> 1a. Użytkownik próbuje zgłosić własny post – system blokuje operację i wyświetla komunikat, że nie można zgłaszać własnych treści. 3a. Post został już wcześniej zgłoszony – system informuje użytkownika, że post jest już zgłoszony i nie zapisuje kolejnego zgłoszenia. |

Tabela 4.35: Scenariusz przypadku użycia: Zgłoszenie posta na forum

| KARTA SCENARIUSZA PRZYPADKU UŻYCIA | |
|------------------------------------|-----------------------------------------------------------------------------------------------------------------------|
| Identyfikator: | PU36 |
| Nazwa: | Przeglądanie komentarzy pod postem |
| Priorytet: | Średni |
| Aktorzy: | Użytkownik niezalogowany, Użytkownik zalogowany |
| Opis: | Użytkownik przegląda komentarze dodane pod wybranym postem na forum z możliwością zmiany kolejności ich wyświetlania. |

| | |
|----------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Warunki wstępne: | Wyświetlany jest szczegółowy widok posta na forum. |
| Warunki końcowe: | Lista komentarzy powiązanych z postem została wyświetlona zgodnie z wybranym kryterium sortowania. |
| Główny przepływ zdarzeń: | <ol style="list-style-type: none"> 1. System pobiera komentarze powiązane z wybranym postem i domyślnie wyświetla je w kolejności od najnowszych. 2. Użytkownik wybiera sposób sortowania komentarzy. 3. System aktualizuje listę komentarzy zgodnie z wybranym kryterium sortowania. 4. Użytkownik przewija listę komentarzy i zapoznaje się z ich treścią. |
| Alternatywne przepływy zdarzeń: | <ol style="list-style-type: none"> 1a. Post nie ma jeszcze komentarzy – system wyświetla informację o braku komentarzy. 1b. Wystąpił błąd podczas pobierania komentarzy – system wyświetla komunikat o błędzie i umożliwia ponowną próbę. |

Tabela 4.36: Scenariusz przypadku użycia: Przeglądanie komentarzy pod postem

Scenariusze przypadków użycia dla panelu użytkownika

| KARTA SCENARIUSZA PRZYPADKU UŻYCIA | |
|------------------------------------|------------------------------------|
| Identyfikator: | PU37 |
| Nazwa: | Dodanie spota w panelu użytkownika |
| Priorytet: | Wysoki |

| | |
|----------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Aktorzy: | Użytkownik zalogowany, Usługa do wyświetlania mapy, Usługa do przechowywania plików w chmurze |
| Opis: | Użytkownik dodaje nowy spot poprzez panel. |
| Warunki wstępne: | Użytkownik jest zalogowany i znajduje się w widoku panelu użytkownika. |
| Warunki końcowe: | Nowy spot został zapisany i jest widoczny na mapie oraz w panelu użytkownika. |
| Główny przepływ zdarzeń: | <ol style="list-style-type: none"> 1. Użytkownik wybiera opcję „Dodaj spota”. 2. Użytkownik uzupełnia podstawowe informacje o specie (nazwa, opis, tagi). 3. Użytkownik wskazuje lokalizację spota na mapie. 4. Użytkownik dodaje zdjęcia/filmy do spota. 5. Użytkownik zapisuje spota. 6. System zapisuje dane spota (oraz pliki w chmurze) i aktualizuje mapę. |
| Alternatywne przepływy zdarzeń: | 3a. Podane dane wejściowe są niepoprawne – system wyświetla komunikat i zaznacza wymagające poprawy pola. |

Tabela 4.37: Scenariusz przypadku użycia: Dodanie spota w panelu użytkownika

| KARTA SCENARIUSZA PRZYPADKU UŻYCIA | |
|------------------------------------|----------------------------------|
| Identyfikator: | PU38 |
| Nazwa: | Przeglądanie profilu użytkownika |
| Priorytet: | Wysoki |
| Aktorzy: | Użytkownik zalogowany |

| | |
|----------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Opis: | Użytkownik przegląda swój profil (lista spotów, media, podstawowe dane). |
| Warunki wstępne: | Użytkownik jest zalogowany. |
| Warunki końcowe: | Wyświetlony jest widok profilu użytkownika wraz z jego zawartością. |
| Główny przepływ zdarzeń: | <ol style="list-style-type: none"> 1. Użytkownik otwiera swój profil. 2. System pobiera dane profilu (informacje podstawowe, spoty, media). 3. System wyświetla dane w odpowiednich sekcjach (spoty, zdjęcia, filmy, komentarze). |
| Alternatywne przepływy zdarzeń: | <p>2a. Wystąpił błąd podczas pobierania danych użytkownika – system wyświetla informację o błędzie.</p> |

Tabela 4.38: Scenariusz przypadku użycia: Przeglądanie profilu użytkownika

| KARTA SCENARIUSZA PRZYPADKU UŻYCIA | |
|------------------------------------|------------------------------------------------------------------------------------|
| Identyfikator: | PU39 |
| Nazwa: | Przeglądanie profilu innego użytkownika |
| Priorytet: | Średni |
| Aktorzy: | Użytkownik niezalogowany |
| Opis: | Użytkownik ogląda profil innego użytkownika (np. z mapy, forum lub społeczności). |
| Warunki wstępne: | Użytkownik jest zalogowany i ma dostęp do odnośnika do profilu innego użytkownika. |

| | |
|----------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Warunki końcowe: | Profil innego użytkownika został wyświetlony. |
| Główny przepływ zdarzeń: | <ol style="list-style-type: none"> 1. Użytkownik wybiera odnośnik do profilu innego użytkownika. 2. System pobiera dane profilu docelowego użytkownika. 3. System wyświetla profil (media, podstawowe informacje). |
| Alternatywne przepływy zdarzeń: | <ol style="list-style-type: none"> <li value="2">2a. Wystąpił błąd podczas pobierania danych użytkownika – system wyświetla informację o błędzie. |

Tabela 4.39: Scenariusz przypadku użycia: Przeglądanie profilu innego użytkownika

| KARTA SCENARIUSZA PRZYPADKU UŻYCIA | |
|-------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Identyfikator: | PU40 |
| Nazwa: | Dodanie użytkownika do znajomych |
| Priorytet: | Średni |
| Aktorzy: | Użytkownik zalogowany |
| Opis: | Użytkownik wysyła lub akceptuje zaproszenie do znajomych. |
| Warunki wstępne: | Użytkownik jest zalogowany i przegląda profil innego użytkownika. |
| Warunki końcowe: | Relacja „znajomy” została utworzona lub zaproszenie czeka na akceptację. |
| Główny przepływ zdarzeń: | <ol style="list-style-type: none"> 1. Użytkownik kliką przycisk „Dodaj do znajomych”. 2. System sprawdza, czy relacja już istnieje. 3. System tworzy nowe zaproszenie. 4. System informuje o statusie o wysłaniu zaproszenia. |

| | |
|----------------------------------------|-------------------------------------------|
| Alternatywne przepływy zdarzeń: | Brak istotnych alternatywnych przepływów. |
|----------------------------------------|-------------------------------------------|

Tabela 4.40: Scenariusz przypadku użycia: Dodanie użytkownika do znajomych

| KARTA SCENARIUSZA PRZYPADKU UŻYCIA | |
|----------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Identyfikator: | PU41 |
| Nazwa: | Przeglądanie społeczności |
| Priorytet: | Średni |
| Aktorzy: | Użytkownik zalogowany |
| Opis: | Użytkownik przegląda społeczności, grupy lub listy znajomych powiązane z aplikacją. |
| Warunki wstępne: | Użytkownik jest zalogowany. |
| Warunki końcowe: | Lista społeczności lub znajomych została wyświetlona. |
| Główny przepływ zdarzeń: | <ol style="list-style-type: none"> 1. Użytkownik przechodzi do sekcji społeczności. 2. System pobiera listę społeczności i znajomych użytkownika. 3. System wyświetla listę z możliwością przechodzenia do profili i czatów. |
| Alternatywne przepływy zdarzeń: | <ol style="list-style-type: none"> <li value="2">2a. Nie udało się pobrać danych – system wyświetla komunikat o błędzie. |

Tabela 4.41: Scenariusz przypadku użycia: Przeglądanie społeczności

| KARTA SCENARIUSZA PRZYPADKU UŻYCIA | |
|----------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Identyfikator: | PU42 |
| Nazwa: | Przeglądanie dodanych spotów |
| Priorytet: | Wysoki |
| Aktorzy: | Użytkownik zalogowany, Usługa do wyświetlania mapy |
| Opis: | Użytkownik przegląda listę/siatkę spotów, które sam dodał. |
| Warunki wstępne: | Użytkownik jest zalogowany i znajduje się w widoku panelu użytkownika lub sekcji „Moje spedy”. |
| Warunki końcowe: | Lista dodanych spotów użytkownika została wyświetlona (np. na mapie i/lub w formie listy). |
| Główny przepływ zdarzeń: | <ol style="list-style-type: none"> Użytkownik przechodzi do sekcji „Moje spedy”. System pobiera listę spotów dodanych przez użytkownika. System wyświetla listę spotów oraz znaczniki na mapie. Użytkownik wybiera spota, aby przejść do jego szczegółów. |
| Alternatywne przepływy zdarzeń: | <p>2a. Użytkownik nie dodał jeszcze żadnego spota – system wyświetla komunikat i proponuje dodanie pierwszego spota.</p> |

Tabela 4.42: Scenariusz przypadku użycia: Przeglądanie dodanych spotów

| KARTA SCENARIUSZA PRZYPADKU UŻYCIA | |
|------------------------------------|---------------------------|
| Identyfikator: | PU43 |
| Nazwa: | Edycja danych użytkownika |
| Priorytet: | Wysoki |
| Aktorzy: | Użytkownik zalogowany |

| | |
|----------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Opis: | Użytkownik modyfikuje swoje dane profilu (np. nazwę, opis, avatar). |
| Warunki wstępne: | Użytkownik jest zalogowany i znajduje się w widoku edycji profilu. |
| Warunki końcowe: | Zaktualizowane dane profilu są zapisane i widoczne w aplikacji. |
| Główny przepływ zdarzeń: | <ol style="list-style-type: none"> 1. Użytkownik otwiera widok edycji profilu. 2. Użytkownik wprowadza zmiany w danych profilu. 3. Użytkownik zapisuje zmiany. 4. System waliduje dane i zapisuje zaktualizowany profil. |
| Alternatywne przepływy zdarzeń: | <p>4a. Dane są niepoprawne lub niekompletne – system wyświetla komunikat o błędzie i zaznacza pola do poprawy.</p> |

Tabela 4.43: Scenariusz przypadku użycia: Edycja danych użytkownika

| KARTA SCENARIUSZA PRZYPADKU UŻYCIA | |
|------------------------------------|---------------------------------------------------------------------------------|
| Identyfikator: | PU44 |
| Nazwa: | Przeglądanie dodanych zdjęć do spotów |
| Priorytet: | Średni |
| Aktorzy: | Użytkownik zalogowany, Usługa do przechowywania plików w chmurze |
| Opis: | Użytkownik przegląda wszystkie zdjęcia powiązane ze spotami, które dodał. |
| Warunki wstępne: | Użytkownik jest zalogowany i znajduje się w sekcji mediów (np. „Moje zdjęcia”). |

| | |
|----------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Warunki końcowe: | Lista lub galeria zdjęć powiązanych ze spotami użytkownika została wyświetlona. |
| Główny przepływ zdarzeń: | <ol style="list-style-type: none"> 1. Użytkownik otwiera sekcję przeglądania zdjęć ze spotów. 2. System pobiera metadane zdjęć z usługi przechowywania plików. 3. System wyświetla galerię zdjęć z podstawowymi informacjami (np. nazwa spota, data dodania). |
| Alternatywne przepływy zdarzeń: | <ol style="list-style-type: none"> 2a. Użytkownik nie dodał jeszcze zdjęć – system wyświetla informację o braku zdjęć. 2b. Nie udało się pobrać danych – system wyświetla komunikat o błędzie. |

Tabela 4.44: Scenariusz przypadku użycia: Przeglądanie dodanych zdjęć do spotów

| KARTA SCENARIUSZA PRZYPADKU UŻYCIA | |
|------------------------------------|----------------------------------------------------------------------------------|
| Identyfikator: | PU45 |
| Nazwa: | Przeglądanie dodanych filmów do spotów |
| Priorytet: | Średni |
| Aktorzy: | Użytkownik zalogowany, Usługa do przechowywania plików w chmurze |
| Opis: | Użytkownik przegląda filmy powiązane ze spotami, które dał. |
| Warunki wstępne: | Użytkownik jest zalogowany i znajduje się w sekcji mediów (np. „Moje filmy”). |
| Warunki końcowe: | Lista lub galeria filmów powiązanych ze spotami użytkownika została wyświetlona. |

| | |
|----------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Główny przepływ zdarzeń: | <ol style="list-style-type: none"> 1. Użytkownik otwiera sekcję przeglądania filmów ze spotów. 2. System pobiera metadane filmów z usługi przechowywania plików. 3. System wyświetla listę/galerię filmów z podstawowymi informacjami. 4. Użytkownik wybiera film, aby go odtworzyć. |
| Alternatywne przepływy zdarzeń: | <ol style="list-style-type: none"> <li value="2">2a. Użytkownik nie dodał jeszcze filmów – system wyświetla informację o braku filmów. <li value="3">3a. Nie udało się pobrać filmów – system wyświetla komunikat o błędzie. |

Tabela 4.45: Scenariusz przypadku użycia: Przeglądanie dodanych filmów do spotów

| KARTA SCENARIUSZA PRZYPADKU UŻYCIA | |
|-------------------------------------------|--------------------------------------------------------------------------|
| Identyfikator: | PU46 |
| Nazwa: | Przeglądanie dodanych komentarzy do spotów |
| Priorytet: | Średni |
| Aktorzy: | Użytkownik zalogowany |
| Opis: | Użytkownik przegląda komentarze dodane do spotów, które sam utworzył. |
| Warunki wstępne: | Użytkownik jest zalogowany i otwiera sekcję komentarzy do swoich spotów. |
| Warunki końcowe: | Lista komentarzy do spotów użytkownika została wyświetlona. |

| | |
|----------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Główny przepływ zdarzeń: | <ol style="list-style-type: none"> 1. Użytkownik przechodzi do sekcji komentarzy do własnych spotów. 2. System pobiera komentarze powiązane ze spotami użytkownika. 3. System wyświetla komentarze (np. w kolejności chronologicznej). |
| Alternatywne przepływy zdarzeń: | <ol style="list-style-type: none"> 2a. Żaden z spotów użytkownika nie ma komentarzy – system wyświetla odpowiednią informację. 3a. Nie udało się pobrać komentarzy – system wyświetla komunikat o błędzie. |

Tabela 4.46: Scenariusz przypadku użycia: Przeglądanie dodanych komentarzy do spotów

4.2 Wymagania ogólne i dziedzinowe

4.3 Wymagania funkcjonalne

4.3.1 Funkcjonalności dla mapy

4.3.2 Funkcjonalności dla chatu

4.3.3 Funkcjonalności dla forum

4.3.4 Funkcjonalności dla konta użytkownika

| KARTA WYMAGANIA | | |
|---------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------|
| Identyfikator: | jednoznaczny symbol np. FO1, FO2 .. | Priorytet: M |
| Nazwa: | Profil użytkownika | |
| Opis: | Jako użytkownik chcę mieć dostęp do strony profilu, aby sprawdzić informacje o swoim koncie. | |
| Kryteria akceptacji: | Użytkownik widzi liczby: znajomych, obserwowanych i obserwujących, a także najpopularniejsze zdjęcia. | |
| Dane wejściowe: | Lista zdjęć oraz liczby: znajomych, obserwujących i obserwowanych. | |
| Warunki początkowe: | Użytkownik jest zalogowany. | |
| Warunki końcowe: | Wyświetlone informacje o profilu. | |
| Sytuacje wyjątkowe: | Błąd połączenia z API; brak danych profilu; brak uprawnień (401/403). | |
| Szczegółły 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 ; drona-rze 2.3 . | |
| Wymagania powiązane: | | |

Tabela 4.47: Profil użytkownika

| KARTA WYMAGANIA | | |
|--------------------------|----------------------------------------------------------------------------------------------------------------------------------------------|--------------|
| Identyfikator: | jednoznaczny symbol np. FO1, FO2 .. | Priorytet: M |
| Nazwa: | Lista dodanych spotów | |
| Opis: | Jako użytkownik chcę sprawdzić listę spotów, które <u>dodałem</u> . | |
| 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 + axios</code> ; prezentacja listy z podstawowymi danymi. | |
| Udziałowiec: | Zespół projektowy 2.1 ; promotor 2.2 ; dronarze 2.3 . | |
| Wymagania powiązane: | | |

Tabela 4.48: 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 axios (POST) z withCredentials. | |
| Udziałowiec: | Zespół projektowy 2.1 ; promotor 2.2 ; droniarze 2.3 . | |
| Wymagania powiązane: | | |

Tabela 4.49: 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 spodem 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 + axios</code> ; prezentacja z miniaturami. | |
| Udziałowiec: | Zespół projektowy 2.1 ; promotor 2.2 ; dronarze 2.3 . | |
| Wymagania powiązane: | | |

Tabela 4.50: 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 spodem 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 + axios</code> ; prezentacja z miniaturami. | |
| Udziałowiec: | Zespół projektowy 2.1 ; promotor 2.2 ; dronarze 2.3 . | |
| Wymagania powiązane: | | |

Tabela 4.51: 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 + axios</code> ; standardowa prezentacja listy. | |
| Udziałowiec: | Zespół projektowy 2.1 ; promotor 2.2 ; drona- rze 2.3 . | |
| Wymagania powiązane: | | |

Tabela 4.52: 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 + axios</code> ; standardowa prezentacja listy. | |
| Udziałowiec: | Zespół projektowy 2.1 ; promotor 2.2 ; drona- rze 2.3 . | |
| Wymagania powiązane: | | |

Tabela 4.53: 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 + axios</code> ; standardowa prezentacja listy. | |
| Udziałowiec: | Zespół projektowy 2.1 ; promotor 2.2 ; dronarze 2.3 . | |
| Wymagania powiązane: | | |

Tabela 4.54: 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 + axios</code> ; prezentacja w zakładkach/kategoriach. | |
| Udziałowiec: | Zespół projektowy 2.1 ; promotor 2.2 ; dronarze 2.3 . | |
| Wymagania powiązane: | | |

Tabela 4.55: 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 + axios</code> ; standardowa prezentacja listy. | |
| Udziałowiec: | Zespół projektowy 2.1 ; promotor 2.2 ; dronarze 2.3 . | |
| Wymagania powiązane: | | |

Tabela 4.56: 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 — aktualizowane dane. | |
| Sytuacje wyjątkowe: | Nieprawidłowe dane formularza; błąd połączenia z API. | |
| Szczegóły implementacji: | Formularz w React; validacja 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 2.1 ; promotor 2.2 ; drona-rze 2.3 . | |
| Wymagania powiązane: | | |

Tabela 4.57: Ustawienia profilu

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

Tabela 4.58: 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 2.1 ; promotor 2.2 ; dronarze 2.3 . | |
| Wymagania powiązane: | | |

Tabela 4.59: Dodawanie do znajomych

4.3.5 Funkcjonalności dla logowania i rejestracji

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

Tabela 4.60: Logowanie i rejestracja

4.3.6 Funkcjonalności dla wyszukiwarki spotów

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

Tabela 4.61: 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 spota, 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 , dronarze 2.3 . | | |
| Wymagania powiązane: | SPXX | | |

Tabela 4.62: Strona główna — zaawansowane filtry

4.3.7 Funkcjonalności dla motywu

| KARTA WYMAGANIA | | | |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------|---|
| Identyfikator: | FOXX | Priorytet: | M |
| Nazwa: | Ustawienia motywu | | |
| Opis: | Jako użytkownik chcę móc zmienić motyw aplikacji. | | |
| Kryteria akceptacji: | Dostępna jest opcja przełączenia motywu na <i>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><html></code> ; | | |
| Udziałowiec: | Zespół projektowy 2.1 , promotor 2.2 , drona-rze 2.3 . | | |
| Wymagania powiązane: | | | |

Tabela 4.63: 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><html></code> (eliminuje mignięcie stylów). | | |
| Udziałowiec: | Zespół projektowy 2.1 , promotor 2.2 , drona-rze 2.3 . | | |
| Wymagania powiązane: | | | |

Tabela 4.64: Zapamiętanie preferencji motywu

| KARTA WYMAGANIA | | | |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------|---|
| Identyfikator: | FOXX | Priorytet: | S |
| Nazwa: | Przełącznik motywów w Sidebar | | |
| Opis: | Jako użytkownik chcę szybko zmieniać motyw bez wchodzenia w ustawienia. | | |
| Kryteria akceptacji: | W Sidebar dostępny jest przełącznik <i>Jasny-/Ciemny</i> ; posiada odpowiednio ikony <i>słońca/księżyca</i> ; zmiana następuje natychmiast. | | |
| Dane wejściowe: | Bieżąca preferencja motywów. | | |
| 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 <i>dark</i> na <i>document.documentElement</i> oraz aktualizuje <i>localStorage (theme = 'dark' 'light')</i> ; brak przeładowania strony. | | |
| Udziałowiec: | Zespół projektowy 2.1 , promotor 2.2 , dronarze 2.3 . | | |
| Wymagania powiązane: | | | |

Tabela 4.65: Szybki przełącznik motywów w interfejsie

4.4 Wymagania pozafunkcjonalne

4.5 Wymagania interfejs z otoczeniem

4.6 Wymagania na środowisko docelowe

Rozdział 5

Projekt

5.1 Wzorce projektowe

5.2 Architektura systemu

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

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

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

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

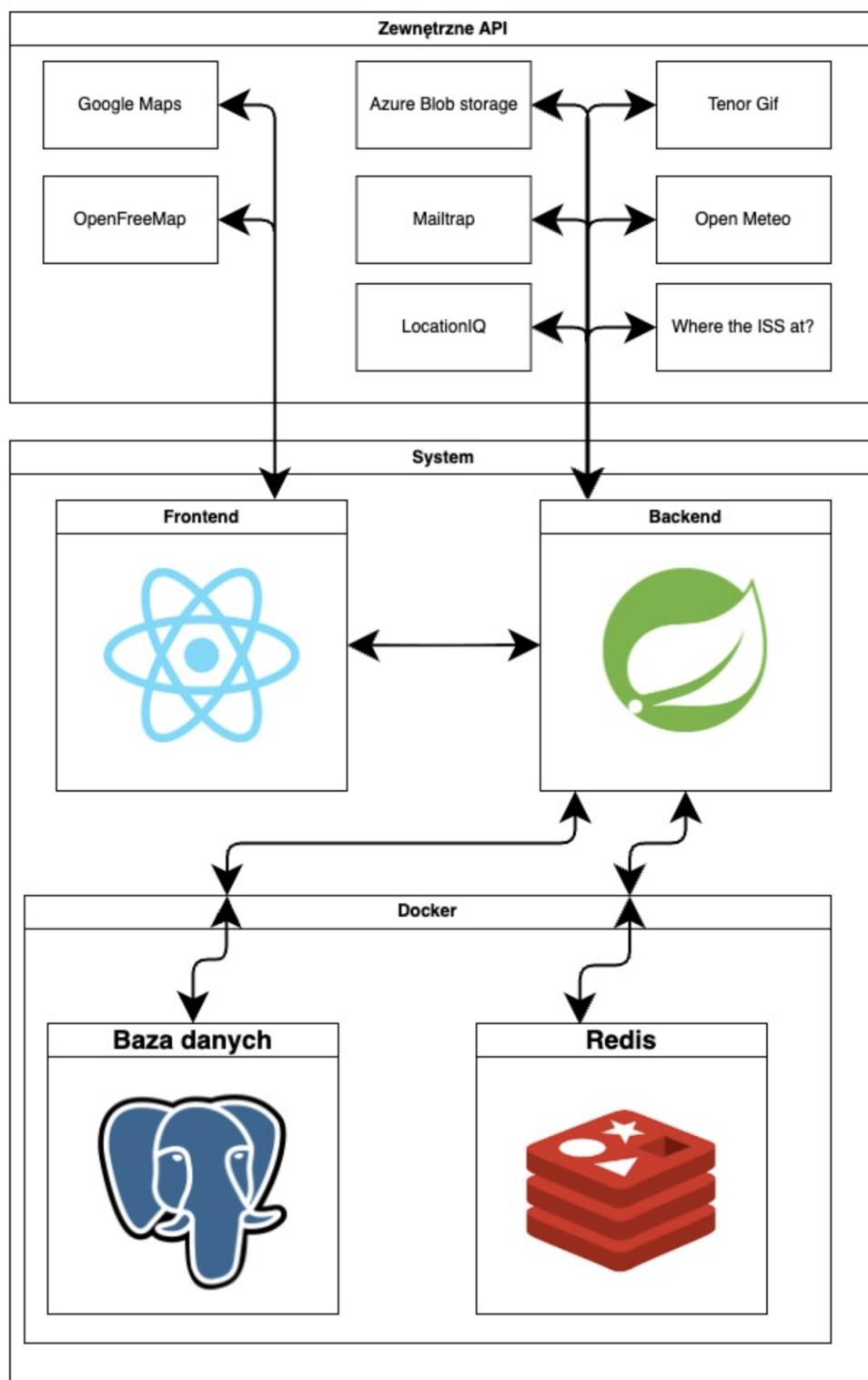
Warstwa wizualna została przygotowana przy użyciu [React](#) w wersji z [TypeScriptem](#) oraz [biblioteki](#) Tailwind CSS, zapewniającej szybkie i wygodne stylowanie komponentów. Z kolei za komunikację oraz logikę biznesową odpowiada [backend](#) oparty na [frameworku](#) Spring Boot, realizujący założenia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

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

5.2.1 Diagram architektury

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

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



Rysunek 5.1: Diagram architektury

5.2.2 Komponenty systemu

System składa się z kilku głównych komponentów, z których każdy pełni ścisłe określona rolę.

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

Szczegółowy wykaz wykorzystywanych zewnętrznych API zamieszczono w rozdziale 3.

- Azure Blob Storage – [3.4](#)
- Mailtrap – [3.5](#)
- LocationIQ – [3.6](#)
- Google Maps – [3.7](#)
- OpenFreeMap – [3.8](#)
- Open Meteo – [3.9](#)
- Tenor Gif – [3.10](#)
- Where the ISS at? – [3.11](#)

5.3 Projekt bazy danych

5.3.1 Model danych

5.3.2 Diagram ERD

5.4 Architektura interfejsu użytkownika

5.4.1 Projekt strony głównej

5.4.2 Projekt panelu logowania

5.4.3 Projekt mapy

5.4.4 Projekt chatu

5.4.5 Projekt forum

5.4.6 Projekt konta użytkownika

Rozdział 6

Przebieg realizacji projektu

6.1 Sprint 1

6.2 Sprint 2

Rozdział 7

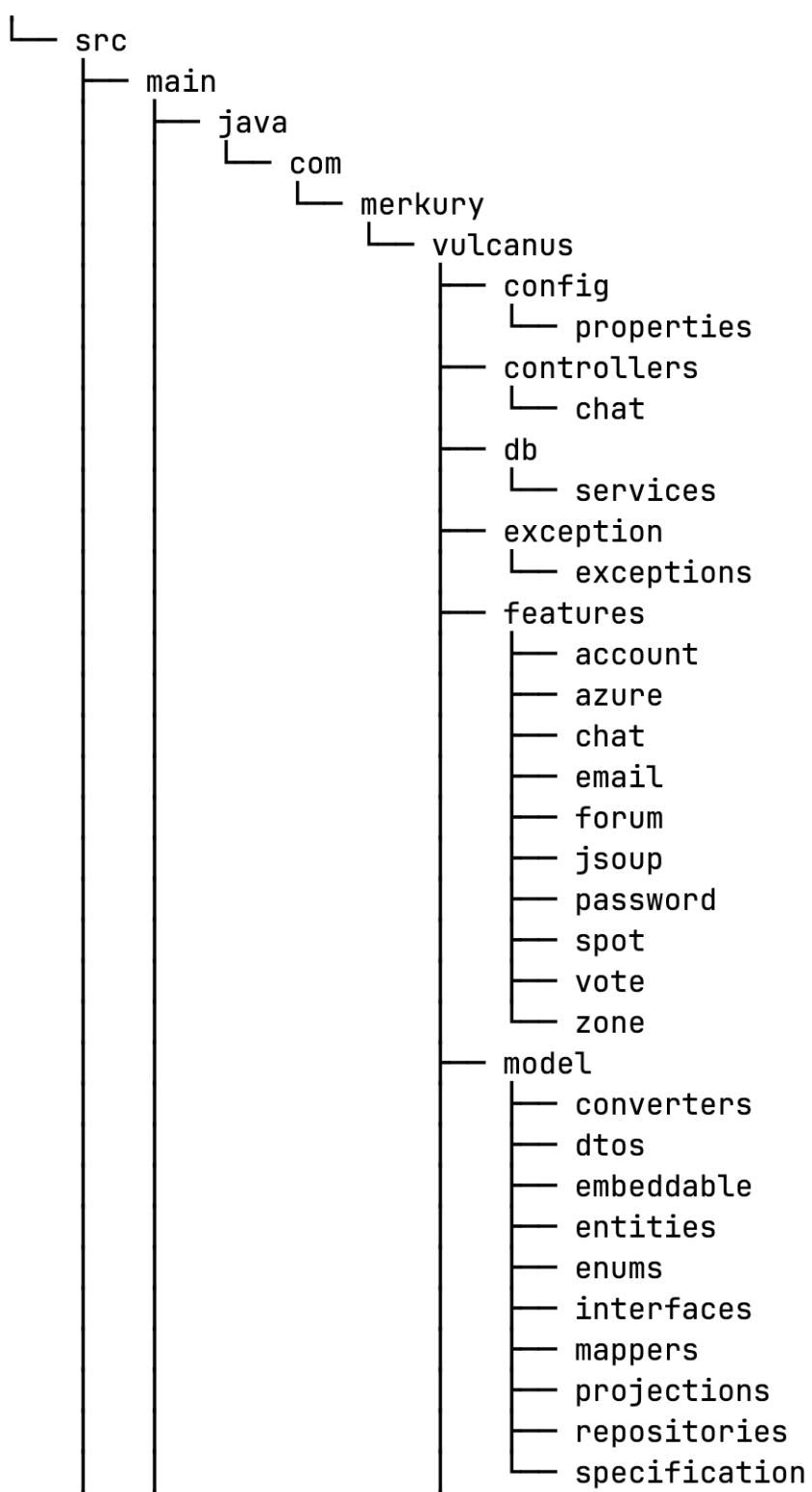
Realizacja Projektu

7.1 Implementacja backendu

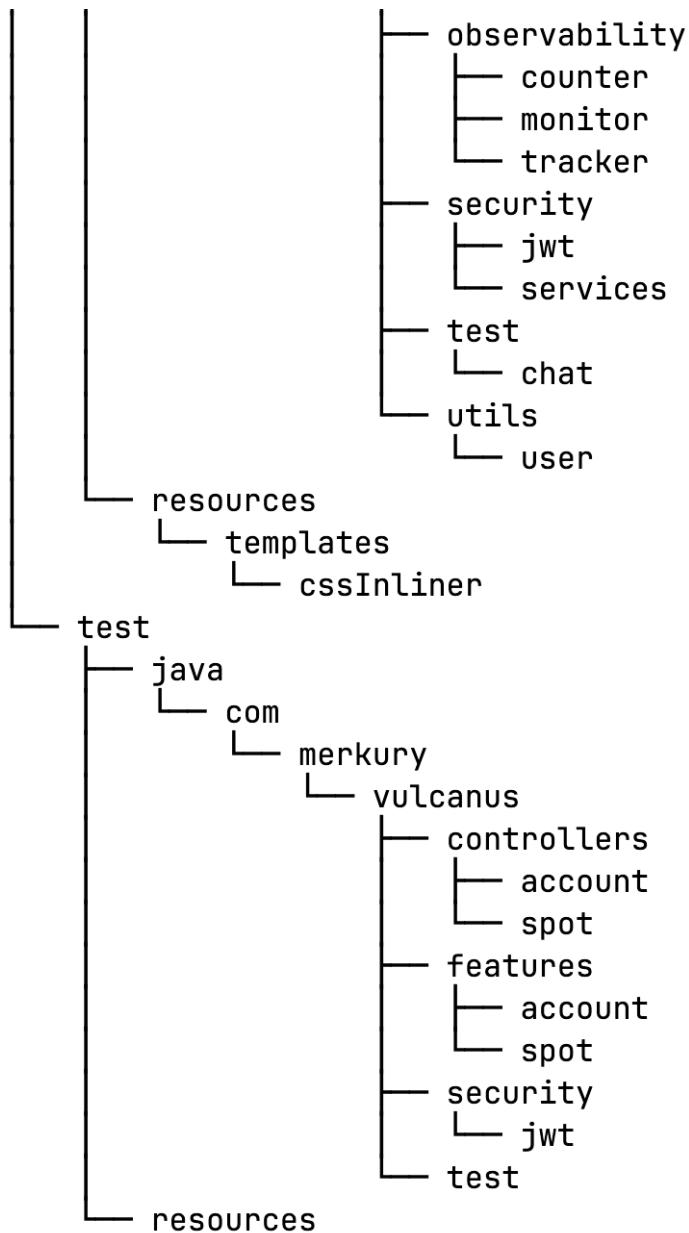
W niniejszym rozdziale przedstawiono strukturę backendu aplikacji, opis implementowanych endpointów, integrację z bazą danych, mechanizmy uwierzytelniania oraz proces konteneryzacji systemu.

7.1.1 Struktura projektu

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



Rysunek 7.1: Struktura katalogów (1)



Rysunek 7.2: Struktura katalogów (2)

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

7.1.2 Endpointy systemu

Projektowany system udostępnia REST-owe API HTTP, za pomocą którego klienci komunikują się z serwerem.

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

metoda HTTP + ścieżka URL

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

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

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

| | |
|-------|-----------------------------------------|
| GET | /user-dashboard/friends/find |
| GET | /user-dashboard/friends/invites |
| PATCH | /user-dashboard/followed |
| GET | /user-dashboard/favorite-spots |
| PATCH | /user-dashboard/favorite-spots |
| POST | /user-dashboard/add-spot-media |
| GET | /user-dashboard/is-spot-favourite |
| GET | /user-dashboard/photos |
| GET | /user-dashboard/comments |
| PATCH | /user-dashboard/settings |
| GET | /user-dashboard/settings |
| GET | /user-dashboard/movies |
| GET | /user-dashboard/photos/{targetUsername} |
| GET | /user-dashboard/add-spot |
| POST | /user-dashboard/add-spot |
| GET | /user-dashboard/add-spot/coordinates |

Tabela 7.1: Zestawienie endpointów: panelu użytkownika

| Zestawienie wszystkich endpointów HTTP modułu spotów | |
|------------------------------------------------------|---------------------------------------|
| Metoda HTTP | Ścieżka |
| GET | /public/spot/gallery |
| GET | /public/spot/gallery-media-position |
| GET | /public/spot/gallery-fullscreen-media |
| GET | /public/spot/current-view |

| | |
|-------|--------------------------------------------------|
| GET | /public/spot/current-view/spot-names |
| GET | /public/spot/{spotId} |
| PATCH | /public/spot/increase-view-count |
| GET | /public/spot/search/map |
| GET | /public/spot/search/list |
| GET | /public/spot/names |
| GET | /public/spot/most-popular |
| GET | /public/spot/search/home-page |
| GET | /public/spot/search/home-page/locations |
| GET | /public/spot/search/home-page/advance |
| GET | /public/spot/get-spot-basic-weather |
| GET | /public/spot/get-spot-detailed-weather |
| GET | /public/spot/get-spot-wind-speeds |
| GET | /public/spot/get-spot-weather-timeline-plot-data |
| PATCH | /public/spot/increase-spot-media-views-count |
| PATCH | /public/spot/edit-spot-media-likes |
| GET | /spot/check-is-spot-media-liked |
| GET | /public/spot/get-spot-time-zone |

Tabela 7.2: Zestawienie endpointów: modułu spotów

| Zestawienie wszystkich endpointów HTTP komentarzy do spotów | |
|-------------------------------------------------------------|--------------------------------------------|
| Metoda HTTP | Ścieżka |
| GET | /public/spot/{spotId}/comments |
| GET | /public/spot/{spotId}/comments/{commentId} |

| | |
|--------|---------------------------------|
| POST | /spot/{spotId}/comments |
| DELETE | /spot/comments/{commentId} |
| PATCH | /spot/comments/{commentId} |
| PATCH | /spot/comments/{commentId}/vote |
| GET | /spot/comments/vote-type |

Tabela 7.3: Zestawienie endpointów: komentarzy do spotów

| Zestawienie wszystkich endpointów HTTP postów forum | |
|-----------------------------------------------------|-------------------------|
| Metoda HTTP | Ścieżka |
| GET | /public/post/{postId} |
| GET | /public/post |
| POST | /post |
| DELETE | /post/{postId} |
| PATCH | /post/{postId} |
| PATCH | /post/{postId}/vote |
| PATCH | /post/{postId}/follow |
| PATCH | /post/{postId}/report |
| GET | /public/categories-tags |

Tabela 7.4: Zestawienie endpointów: postów forum

| Zestawienie wszystkich endpointów HTTP komentarzy forum | |
|---------------------------------------------------------|---------|
| Metoda HTTP | Ścieżka |

| | |
|--------|--------------------------------------|
| GET | /public/post/{postId}/comments |
| GET | /public/comments/{commentId}/replies |
| POST | /post/{postId}/comments |
| DELETE | /post/comments/{commentId} |
| PATCH | /post/comments/{commentId} |
| PATCH | /post/comments/{commentId}/vote |
| PATCH | /post/comments/{commentId}/report |
| POST | /comments/{commentId}/replies |

Tabela 7.5: Zestawienie endpointów: komentarzy forum

| Zestawienie wszystkich endpointów HTTP konta użytkownika i autoryzacji | |
|------------------------------------------------------------------------|----------------------------------|
| Metoda HTTP | Ścieżka |
| POST | /public/account/register |
| POST | /public/account/login |
| GET | /account/login-success |
| POST | /public/account/forgot-password |
| POST | /public/account/set-new-password |
| GET | /account/check |

Tabela 7.6: Zestawienie endpointów: konta użytkownika i autoryzacji

Zestawienie wszystkich endpointów HTTP integracji GIF-ów (Tenor)

| Metoda HTTP | Ścieżka |
|----------------|----------------|
| GET | /gifs/trending |
| GET | /gifs/search |

Tabela 7.7: Zestawienie endpointów: integracji GIF-ów

| Zestawienie wszystkich endpointów HTTP modułu czatu | |
|-----------------------------------------------------|---------------------------------------|
| Metoda HTTP | Ścieżka |
| GET | /chats/{chatId}/messages |
| GET | /chats/user-chats |
| POST | /chats/get-or-create-private-chat |
| POST | /chats/{chatId}/send-files |
| POST | /chats/create/group |
| PATCH | /chats/{chatId} |
| GET | /chats/group-chat/add/search/{chatId} |
| PUT | /chats/add/users/{chatId} |

Tabela 7.8: Zestawienie endpointów: modułu czatu

Panel użytkownika

| KARTA ENDPOINTU API | |
|-----------------------|-----------------------------------------------------|
| Identyfikator: | EP01 |
| Ścieżka: | /public/user-dashboard/profile/{targetUsername} |
| Nazwa: | Pobierz profil innego użytkownika (widok publiczny) |

| | |
|-----------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Parametry wejściowe: | <ul style="list-style-type: none"> • targetUsername: String (nazwa użytkownika w ścieżce URL) |
| Query params: | Brak |
| Kod(y) statusu odpowiedzi: | 200 OK, 404 Not Found |
| Dane zwracane: | <ul style="list-style-type: none"> • profile: UserProfileDto, zawiera: <ul style="list-style-type: none"> – username: String – profilePhoto: String (URL) – followersCount: Integer – followedCount: Integer – friendsCount: Integer – photosCount: Integer – mostPopularPhotos: List<ImageDto> • friendStatus: UserFriendStatus • isFollowing: Boolean • isOwnProfile: Boolean |

Tabela 7.9: Karta endpointu: /public/user-dashboard/profile/{targetUsername}

| KARTA ENDPOINTU API | |
|-----------------------|---------------------------------------------|
| Identyfikator: | EP02 |
| Ścieżka: | /user-dashboard/favorite-spots |
| Nazwa: | Pobierz listę ulubionych spotów użytkownika |

| | |
|-----------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Parametry wejściowe: | Brak |
| Query params: | <ul style="list-style-type: none"> • type: FavoriteSpotsListType (typ listy: ulubione, odwiedzone oraz do odwiedzenia) • page: Integer (opcjonalnie, domyślnie 0) • size: Integer (opcjonalnie, domyślnie 10) |
| Kod(y) statusu odpowiedzi: | 200 OK, 401 Unauthorized |
| Dane zwracane: | <ul style="list-style-type: none"> • items: List<FavoriteSpotDto>, gdzie każdy element zawiera: <ul style="list-style-type: none"> – id: Long – name: String – country: String – city: String – description: String – rating: Double – viewsCount: Integer – imageUrl: String – type: FavoriteSpotsListType – coords: SpotCoordinatesDto – tags: Set<SpotTagDto> • hasNext: boolean |

Tabela 7.10: Karta endpointu: /user-dashboard/favorite-spots

| KARTA ENDPOINTU API | |
|-----------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Identyfikator: | EP03 |
| Ścieżka: | /user-dashboard/photos |
| Nazwa: | Pobierz posortowane zdjęcia użytkownika |
| Parametry wejściowe: | Brak |
| Query params: | <ul style="list-style-type: none"> • type: DateSortType (typ sortowania po dacie) • from: LocalDate (opcjonalnie, data od) • to: LocalDate (opcjonalnie, data do) • page: Integer (opcjonalnie, domyślnie 0) • size: Integer (opcjonalnie, domyślnie 20) |
| Kod(y) statusu odpowiedzi: | 200 OK, 400 Bad Request, 401 Unauthorized |
| Dane zwracane: | <ul style="list-style-type: none"> • items: List<DatedMediaGroupDto>, gdzie: <ul style="list-style-type: none"> – date: LocalDate (data grupy) – media: List<MediaDto>, każdy element: <ul style="list-style-type: none"> * src: String (URL) * heartsCount: Integer * viewsCount: Integer * id: Long • hasNext: boolean |

Tabela 7.11: Karta endpointu: /user-dashboard/photos

| KARTA ENDPOINTU API | |
|-----------------------|------|
| Identyfikator: | EP04 |

| | |
|-----------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Ścieżka: | /user-dashboard/add-spot |
| Nazwa: | Pobierz listę spotów dodanych przez użytkownika |
| Parametry wejściowe: | Brak |
| Query params: | <ul style="list-style-type: none"> • page: Integer (opcjonalnie, domyślnie 0) • size: Integer (opcjonalnie, domyślnie 20) |
| Kod(y) statusu odpowiedzi: | 200 OK, 401 Unauthorized |
| Dane zwracane: | <ul style="list-style-type: none"> • items: List<AddSpotDto>, każdy element: <ul style="list-style-type: none"> – id: Long – name: String – description: String – country: String – region: String – city: String – street: String – borderPoints: List<BorderPoint> (x, y) – firstPhotoUrl: String • hasNext: boolean |

Tabela 7.12: Karta endpointu: /user-dashboard/add-spot

| KARTA ENDPOINTU API | |
|-----------------------|------|
| Identyfikator: | EP05 |

| | |
|-----------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Ścieżka: | /user-dashboard/add-spot |
| Nazwa: | Dodaj nowy spot użytkownika |
| Parametry wejściowe: | <ul style="list-style-type: none"> spot: String (część multipart, JSON z danymi nowego spotu) media: List<MultipartFile> (część multipart, pliki multimedialne spota) |
| Query params: | Brak |
| Kod(y) statusu odpowiedzi: | 200 OK, 404 Not Found, 401 Unauthorized |
| Dane zwracane: | Brak (pusta odpowiedź) |

Tabela 7.13: Karta endpointu: /user-dashboard/add-spot

Spoty i pogoda

| KARTA ENDPOINTU API | |
|-----------------------------|-----------------------------------------|
| Identyfikator: | EP06 |
| Ścieżka: | /public/spot/gallery |
| Nazwa: | Pobierz stronę galerii mediów dla spota |
| Parametry wejściowe: | Brak |

| | |
|-----------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Query params: | <ul style="list-style-type: none"> • spotId: Long (identyfikator spota) • mediaType: String (typ plików, wartość enum GenericMediaType, PHOTO, VIDEO) • sorting: String (kryterium sortowania, po dacie / popularności) • page: Integer (numer strony, domyślnie 0) • size: Integer (rozmiar strony, domyślnie 6) |
| Kod(y) statusu odpowiedzi: | 200 OK, 404 Not Found |
| Dane zwracane: | <ul style="list-style-type: none"> • Page<SpotSidebarMediaGalleryDto>: stronicowana lista elementów galerii, gdzie każdy element zawiera: <ul style="list-style-type: none"> – id: Long (identyfikator media) – url: String (URL pliku) – mediaType: GenericMediaType (typ pliku) |

Tabela 7.14: Karta endpointu: /public/spot/gallery

| KARTA ENDPOINTU API | |
|-----------------------------|----------------------------------------------|
| Identyfikator: | EP07 |
| Ścieżka: | /public/spot/current-view |
| Nazwa: | Pobierz listę spotów w aktualnym widoku mapy |
| Parametry wejściowe: | Brak |

| | |
|-----------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Query params: | <ul style="list-style-type: none"> • swLng: double (długość geograficzna lewego dolnego rogu) • swLat: double (szerokość geograficzna lewego dolnego rogu) • neLng: double (długość geograficzna prawego górnego rogu) • neLat: double (szerokość geograficzna prawego górnego rogu) • name: String (fragment nazwy spotu, domyślnie pusty) • sorting: String (tryb sortowania, domyślnie none) • ratingFrom: double (minimalna ocena, domyślnie 0.0) • page: Integer (numer strony, domyślnie 0) |
| Kod(y) statusu odpowiedzi: | 200 OK |
| Dane zwracane: | <ul style="list-style-type: none"> • Page<SearchSpotDto>: stronicowana lista spotów, gdzie każdy element zawiera: <ul style="list-style-type: none"> – id: Long (identyfikator spota) – name: String (nazwa spota) – rating: Double (0–5) – ratingCount: Integer (liczba ocen) – firstPhoto: String (URL pierwszego zdjęcia) – tags: Set<SpotTagDto> (tagi spota) – centerPoint: BorderPoint (środek obszaru spota) |

Tabela 7.15: Karta endpointu: /public/spot/current-view

KARTA ENDPOINTU API

| | |
|-----------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Identyfikator: | EP08 |
| Ścieżka: | /public/spot/get-spot-basic-weather |
| Nazwa: | Pobierz podstawowe informacje pogodowe dla wskazanej lokalizacji |
| Parametry wejściowe: | Brak |
| Query params: | <ul style="list-style-type: none"> • latitude: double (szerokość geograficzna) • longitude: double (długość geograficzna) |
| Kod(y) statusu odpowiedzi: | 200 OK |
| Dane zwracane: | <ul style="list-style-type: none"> • temperature: Double (temperatura) • weatherCode: int (kod warunków pogodowych) • windSpeed: Double (prędkość wiatru) • isDay: boolean (czy jest dzień) |

Tabela 7.16: Karta endpointu: /public/spot/get-spot-basic-weather

| KARTA ENDPOINTU API | |
|-----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Identyfikator: | EP09 |
| Ścieżka: | /public/spot/get-spot-detailed-weather |
| Nazwa: | Pobierz szczegółowe informacje pogodowe dla wskazanej lokalizacji |
| Parametry wejściowe: | Brak |
| Query params: | <ul style="list-style-type: none"> • latitude: double (szerokość geograficzna) • longitude: double (długość geograficzna) |

| | |
|-------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Kod(y) statusu odpowiedzi: | 200 OK |
| Dane zwracane: | <ul style="list-style-type: none"> • temperature: Double (temperatura) • weatherCode: int (kod warunków pogodowych) • precipitationProbability: Double (prawdopodobieństwo opadów) • dewPoint: Double (punkt rosy) • relativeHumidity: Double (wilgotność względna) • isDay: boolean (czy jest dzień) • uvIndexMax: Double (maksymalny indeks UV) |

Tabela 7.17: Karta endpointu: /public/spot/get-spot-detailed-weather

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

| | |
|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Dane zwracane: | <ul style="list-style-type: none"> • windSpeeds100m: Double (prędkość wiatru na wysokości 100 metrów) • windSpeeds200m: Double (prędkość wiatru na wysokości 200 metrów) • windSpeeds300m: Double (prędkość wiatru na wysokości 300 metrów) • windSpeeds500m: Double (prędkość wiatru na wysokości 500 metrów) • windSpeeds750m: Double (prędkość wiatru na wysokości 750 metrów) • windSpeeds1000m: Double (prędkość wiatru na wysokości 1000 metrów) |
|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Tabela 7.18: Karta endpointu: /public/spot/get-spot-wind-speeds

Wyszukiwarka spotów

| KARTA ENDPOINTU API | |
|-----------------------------------|---------------------------------------|
| Identyfikator: | EP11 |
| Ścieżka: | /public/spot/most-popular |
| Nazwa: | Pobierz 18 najpopularniejszych spotów |
| Parametry wejściowe: | Brak |
| Query params: | Brak |
| Kod(y) statusu odpowiedzi: | 200 OK |

| | |
|-----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Dane zwracane: | <ul style="list-style-type: none"> • items: List<TopRatedSpotDto> każdy element zawiera: <ul style="list-style-type: none"> – id: Long (identyfikator spota) – name: String (nazwa spota) – city: String (miasto, w którym znajduje się spot) – imageUrl: String (URL zdjęcia spota) |
|-----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Tabela 7.19: Karta endpointu: /public/spot/most-popular

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

| | |
|-----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Dane zwracane: | <ul style="list-style-type: none"> • items: List<HomePageSpotDto> (lista znalezionych spotów), każdy element: <ul style="list-style-type: none"> – id: Long (identyfikator spota) – name: String (nazwa spota) – rating: Double (średnia ocena spota) – ratingCount: Integer (liczba ocen spota) – firstPhoto: String (URL pierwszego zdjęcia spota) – tags: Set<SpotTagDto> (zestaw tagów przypisanych do spota) – centerPoint: BorderPoint (punkt centralny obszaru spota) – city: String (miasto, w którym znajduje się spot) – distanceToUser: Double (odległość od lokalizacji użytkownika, jeśli dostępna) • hasNext: boolean (czy istnieje kolejna strona wyników) |
|-----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Tabela 7.20: Karta endpointu: /public/spot/search/home-page

| KARTA ENDPOINTU API | |
|-----------------------------|---------------------------------------------------------------------------|
| Identyfikator: | EP13 |
| Ścieżka: | /public/spot/search/home-page/locations |
| Nazwa: | Pobierz listę podpowiedzi lokalizacji dla wyszukiwarki na stronie głównej |
| Parametry wejściowe: | Brak |

| | |
|-----------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Query params: | <ul style="list-style-type: none"> • query: String (frazą wyszukiwania) • type: String (typ lokalizacji, kraj/region/miasto) |
| Kod(y) statusu odpowiedzi: | 200 OK |
| Dane zwracane: | <ul style="list-style-type: none"> • List<String>: lista podpowiedzi (nazwy lokalizacji) |

Tabela 7.21: Karta endpointu: /public/spot/search/home-page/locations

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

| | |
|-------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Kod(y) statusu odpowiedzi: | 200 OK |
| Dane zwracane: | <ul style="list-style-type: none"> • items: List<HomePageSpotDto> (lista znalezionych spotów), każdy element: <ul style="list-style-type: none"> – id: Long (identyfikator spota) – name: String (nazwa spota) – rating: Double (średnia ocena spota) – ratingCount: Integer (liczba ocen spota) – firstPhoto: String (URL pierwszego zdjęcia spota) – tags: Set<SpotTagDto> (zestaw tagów przypisanych do spota) – centerPoint: BorderPoint (punkt centralny obszaru spota) – city: String (miasto, w którym znajduje się spot) – distanceToUser: Double (odległość od lokalizacji użytkownika, jeśli dostępna) • hasNext: boolean (czy istnieje kolejna strona wyników) |

Tabela 7.22: Karta endpointu: /public/spot/search/home-page/advance

Komentarze do spotów

| KARTA ENDPOINTU API | |
|-----------------------|------------------------------------------------------------|
| Identyfikator: | EP15 |
| Ścieżka: | /public/spot/{spotId}/comments |
| Nazwa: | Pobierz stronicowaną listę komentarzy dla wskazanego spota |

| | |
|-----------------------------------|--------------------------------------------------------------------------------------------------------------------------|
| Parametry wejściowe: | <ul style="list-style-type: none"> • spotId: Long (identyfikator spota w ścieżce URL) |
| Query params: | <ul style="list-style-type: none"> • page: Integer (numer strony, domyślnie 0; rozmiar strony = 2) |
| Kod(y) statusu odpowiedzi: | 200 OK, 404 Not Found, 401 Unauthorized |

| | |
|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Dane zwracane: | <ul style="list-style-type: none"> • Page<SpotCommentDto>: stronicowana lista komentarzy dla danego spota, każdy element zawiera: <ul style="list-style-type: none"> – id: Long (identyfikator komentarza) – author: SpotCommentAuthorDto (dane autora komentarza) – text: String (treść komentarza) – rating: Double (ocena spota wystawiona w komentarzu, 0–5) – upvotes: Integer (liczba głosów pozytywnych na komentarz) – downvotes: Integer (liczba głosów negatywnych na komentarz) – publishDate: LocalDateTime (data i godzina publikacji komentarza) – isUpVoted: Boolean (czy bieżący użytkownik oddał głos w górę na ten komentarz) – isDownVoted: Boolean (czy bieżący użytkownik oddał głos w dół na ten komentarz) – numberOfMedia: Integer (łączna liczba dołączonych plików multimedialnych) – mediaList: List<SpotCommentMediaDto> (lista pierwszych plików komentarza) |
|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Tabela 7.23: Karta endpointu: /public/spot/{spotId}/comments

KARTA ENDPOINTU API

| | |
|-----------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Identyfikator: | EP16 |
| Ścieżka: | /public/spot/{spotId}/comments/{commentId} |
| Nazwa: | Pobierz pełną listę mediów powiązanych z komentarzem |
| Parametry wejściowe: | <ul style="list-style-type: none"> spotId: Long (identyfikator spota w ścieżce URL) commentId: Long (identyfikator komentarza w ścieżce URL) |
| Query params: | Brak |
| Kod(y) statusu odpowiedzi: | 200 OK, 404 Not Found |
| Dane zwracane: | <ul style="list-style-type: none"> List<SpotCommentMediaDto>: pełna lista mediów powiązanych z komentarzem, każdy element: <ul style="list-style-type: none"> id: Long (identyfikator pliku multimedialnego) url: String (URL pliku, używany do pobrania/wyświetlenia) genericMediaType: GenericMediaType (typ pliku, PHOTO lub VIDEO) |

Tabela 7.24: Karta endpointu: /public/spot/{spotId}/comments/{commentId}

| KARTA ENDPOINTU API | |
|-----------------------|------------------------------------------|
| Identyfikator: | EP17 |
| Ścieżka: | /spot/{spotId}/comments |
| Nazwa: | Dodaj nowy komentarz do wskazanego spota |

| | |
|-----------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Parametry wejściowe: | <ul style="list-style-type: none"> • spotId: Long (identyfikator spota w ścieżce URL) • body: SpotCommentAddDto (dane nowego komentarza), zawiera: <ul style="list-style-type: none"> – text: String (treść komentarza) – rating: Double (ocena spota w komentarzu, zakres 0–5) – mediaFiles: List<MultipartFile> (lista załączonych plików, zdjęcia/filmy) |
| Query params: | Brak |
| Kod(y) statusu odpowiedzi: | 201 Created, 404 Not Found, 401 Unauthorized, 422 Unprocessable Entity |
| Dane zwracane: | Brak (pusta odpowiedź) |

Tabela 7.25: Karta endpointu: /spot/{spotId}/comments

| KARTA ENDPOINTU API | |
|-----------------------------|-----------------------------------------------------------------------------------------------------------------------|
| Identyfikator: | EP18 |
| Ścieżka: | /spot/comments/{commentId}/vote |
| Nazwa: | Oddaj głos na komentarz (góra/dół) |
| Parametry wejściowe: | <ul style="list-style-type: none"> • commentId: Long (identyfikator komentarza w ścieżce URL) |
| Query params: | <ul style="list-style-type: none"> • isUpvote: boolean (true = głos w góre, false = głos w dół) |

| | |
|-------------------------------------------|----------------------------------------------------------------------|
| Kod(y) statusu odpowiedzi: | 200 OK, 401 Unauthorized, 404 Not Found, 409 Conflict, 403 Forbidden |
| Dane zwracane: | Brak (pusta odpowiedź) |

Tabela 7.26: Karta endpointu: /spot/comments/{commentId}/vote

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

Tabela 7.27: Karta endpointu: /spot/comments/vote-type

Forum – posty

| KARTA ENDPOINTU API | |
|-----------------------------------|-------------------------------------------------------------------------------------------------------------|
| Identyfikator: | EP20 |
| Ścieżka: | /public/post/{postId} |
| Nazwa: | Pobierz szczegółowe informacje o poście |
| Parametry wejściowe: | <ul style="list-style-type: none"> • postId: Long (identyfikator posta w ścieżce URL) |
| Query params: | Brak |
| Kod(y) statusu odpowiedzi: | 200 OK, 404 Not Found |

| | |
|------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>Dane zwracane:</p> | <ul style="list-style-type: none"> • PostDetailsDto, zawiera: – id: Long (identyfikator posta) – title: String (tytuł posta) – content: String (pełna treść posta) – category: ForumCategoryDto (kategoria forum, do której należy post) – tags: List<ForumTagDto> (lista tagów przypisanych do posta) – author: AuthorDto (dane autora posta) – isAuthor: Boolean (czy bieżący użytkownik jest autorem posta) – isFollowed: Boolean (czy bieżący użytkownik obserwuje ten post) – publishDate: LocalDateTime (data i godzina publikacji posta) – views: Integer (liczba wyświetleń posta) – upVotes: Integer (liczba głosów w górę na post) – downVotes: Integer (liczba głosów w dół na post) – isUpVoted: Boolean (czy bieżący użytkownik oddał głos w górę na post) – isDownVoted: Boolean (czy bieżący użytkownik oddał głos w dół na post) – commentsCount: Integer (łączna liczba komentarzy pod postem) |
|------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Tabela 7.28: Karta endpointu: /public/post/{postId}

| KARTA ENDPOINTU API | |
|-----------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Identyfikator: | EP21 |
| Ścieżka: | /post |
| Nazwa: | Dodaj nowy post na forum |
| Parametry wejściowe: | <ul style="list-style-type: none"> • body: PostDto (dane nowego posta), zawiera: <ul style="list-style-type: none"> – title: String (tytuł posta) – content: String (pełna treść posta) – category: String (nazwa kategorii forum, do której ma trafić post) – tags: List<String> (lista nazw tagów przypisanych do posta) |
| Query params: | Brak |
| Kod(y) statusu odpowiedzi: | 201 Created, 400 Bad Request, 404 Not Found, 409 Conflict, 422 Unprocessable Entity |
| Dane zwracane: | Brak (pusta odpowiedź) |

Tabela 7.29: Karta endpointu: /post

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

| | |
|-----------------------------------|-------------------------------------------------|
| Query params: | Brak |
| Kod(y) statusu odpowiedzi: | 204 No Content, 401 Unauthorized, 404 Not Found |
| Dane zwracane: | Brak (pusta odpowiedź) |

Tabela 7.30: Karta endpointu: /post/{postId}

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

Tabela 7.31: Karta endpointu: /post/{postId}/vote

| KARTA ENDPOINTU API | |
|-----------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Identyfikator: | EP24 |
| Ścieżka: | /public/categories-tags |
| Nazwa: | Pobierz listę wszystkich kategorii i tagów forum |
| Parametry wejściowe: | Brak |
| Query params: | Brak |
| Kod(y) statusu odpowiedzi: | 200 OK |
| Dane zwracane: | <ul style="list-style-type: none"> • ForumCategoriesAndTagsDto (zestaw kategorii i tagów forum), zawiera: <ul style="list-style-type: none"> – categories: List<ForumCategoryDto> (lista dostępnych kategorii), gdzie każdy element zawiera: <ul style="list-style-type: none"> * id: Long (identyfikator kategorii) * name: String (nazwa kategorii) * description: String (opis kategorii) * colour: String (kolor kategorii) – tags: List<ForumTagDto> (lista dostępnych tagów), gdzie każdy element zawiera: <ul style="list-style-type: none"> * id: Long (identyfikator tagu) * name: String (nazwa tagu) |

Tabela 7.32: Karta endpointu: /public/categories-tags

Forum – komentarze do postów

| KARTA ENDPOINTU API | |
|-----------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Identyfikator: | EP25 |
| Ścieżka: | /public/post/{postId}/comments |
| Nazwa: | Pobierz stronicowaną listę komentarzy posta |
| Parametry wejściowe: | <ul style="list-style-type: none"> • postId: Long (identyfikator posta w ścieżce URL) |
| Query params: | <ul style="list-style-type: none"> • page: Integer (numer strony, domyślnie 0) • size: Integer (liczba komentarzy na stronie, domyślnie 10) • sortBy: PostCommentSortField (pole sortowania, domyślnie PUBLISH_DATE) • sortDirection: SortDirection (kierunek sortowania, domyślnie DESC) |
| Kod(y) statusu odpowiedzi: | 200 OK, 404 Not Found |

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

Tabela 7.33: Karta endpointu: /public/post/{postId}/comments

| KARTA ENDPOINTU API | |
|-----------------------|-------------------------------|
| Identyfikator: | EP26 |
| Ścieżka: | /post/{postId}/comments |
| Nazwa: | Dodaj nowy komentarz do posta |

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

Tabela 7.34: Karta endpointu: /post/{postId}/comments

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

| | |
|-----------------------|------------------------|
| Dane zwracane: | Brak (pusta odpowiedź) |
|-----------------------|------------------------|

Tabela 7.35: Karta endpointu: /post/comments/{commentId}

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

Tabela 7.36: Karta endpointu: /post/comments/{commentId}/vote

| KARTA ENDPOINTU API | |
|-----------------------|-------------------------------|
| Identyfikator: | EP29 |
| Ścieżka: | /comments/{commentId}/replies |
| Nazwa: | Dodaj odpowiedź na komentarz |

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

Tabela 7.37: Karta endpointu: /comments/{commentId}/replies

Konto użytkownika – rejestracja, logowanie, hasło

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

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

Tabela 7.38: Karta endpointu: /public/account/register

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

Tabela 7.39: Karta endpointu: /public/account/login

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

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

Tabela 7.40: Karta endpointu: /public/account/forgot-password

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

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

Tabela 7.41: Karta endpointu: /public/account/set-new-password

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

Tabela 7.42: Karta endpointu: /account/check

| KARTA ENDPOINTU API | |
|-----------------------|------|
| Identyfikator: | EP35 |

| | |
|-----------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Ścieżka: | /account/login-success |
| Nazwa: | Obsłuż użytkownika zalogowanego przez OAuth2 i przekieruj go do aplikacji |
| Parametry wejściowe: | <ul style="list-style-type: none"> Brak klasycznego body – endpoint wywoływany jest jako redirect callback po poprawnym logowaniu przez dostawcę OAuth2. Kontekst użytkownika przekazywany jest w obiekcie <code>OAuth2AuthenticationToken</code>. |
| Query params: | Brak |
| Kod(y) statusu odpowiedzi: | 302 Found (redirect), 404 Not Found, 409 Conflict, 422 Unprocessable Entity, 500 Internal Server Error |
| Dane zwracane: | <ul style="list-style-type: none"> Przekierowanie użytkownika na stronę frontendową skonfigurowaną w <code>UrlsProperties.afterLoginPageUrl</code>. |

Tabela 7.43: Karta endpointu: /account/login-success

GIF-y (Tenor) – integracja czatu

| KARTA ENDPOINTU API | |
|-----------------------------|-----------------------------------------------------|
| Identyfikator: | EP36 |
| Ścieżka: | /gifs/trending |
| Nazwa: | Pobierz listę trendujących kategorii GIF-ów z Tenor |
| Parametry wejściowe: | Brak |
| Query params: | Brak |

| | |
|-------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Kod(y) statusu odpowiedzi: | 200 OK, 500 Internal Server Error |
| Dane zwracane: | <ul style="list-style-type: none"> • items: List<TenorGifCategoryDto>, każdy element zawiera: <ul style="list-style-type: none"> – searchTerm: String (frazą wyszukiwania powiązana z kategorią) – path: String (ścieżka kategorii w Tenor) – gifUrl: String (URL GIF-a) |

Tabela 7.44: Karta endpointu: /gifs/trending

| KARTA ENDPOINTU API | |
|-------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Identyfikator: | EP37 |
| Ścieżka: | /gifs/search |
| Nazwa: | Wyszukaj GIF-y po frazie tekstowej |
| Parametry wejściowe: | Brak |
| Query params: | <ul style="list-style-type: none"> • searchPhrase: String (frazą wyszukiwania) • next: String (token paginacji zwrócony z poprzedniego wywołania; dla pierwszego zapytania może być pusty) |
| Kod(y) statusu odpowiedzi: | 200 OK, 500 Internal Server Error |

| | |
|-----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Dane zwracane: | <ul style="list-style-type: none"> • body: TenorGifSearchWrapperDto (wyniki wyszukiwania GIF-ów), zawiera: <ul style="list-style-type: none"> – gifs: List<TenorGifSearchDto> (lista pasujących GIF-ów), każdy element: <ul style="list-style-type: none"> * url: String (URL GIF-a) – next: String (token do pobrania kolejnej strony wyników) |
|-----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Tabela 7.45: Karta endpointu: /gifs/search

Czat – REST API

| KARTA ENDPOINTU API | |
|-----------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Identyfikator: | EP38 |
| Ścieżka: | /chats/{chatId}/messages |
| Nazwa: | Pobierz stronicowane wiadomości dla wybranego czatu |
| Parametry wejściowe: | <ul style="list-style-type: none"> • chatId: Long (identyfikator czatu) |
| Query params: | <ul style="list-style-type: none"> • pageParam: Integer (numer strony wiadomości, domyślnie 1 – pierwsza strona po wstępnym pobraniu) • numberOfMessagesPerPage: Integer (liczba wiadomości na stronę, domyślnie 20) |
| Kod(y) statusu odpowiedzi: | 200 OK, 401 Unauthorized, 404 Not Found |

| | |
|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Dane zwracane: | <ul style="list-style-type: none"> • body: ChatMessageDtoSlice, zawiera: <ul style="list-style-type: none"> – messages: List<ChatMessageDto>, każdy element: <ul style="list-style-type: none"> * id: Long (identyfikator wiadomości) * sender: ChatMessageSenderDto (dane nadawcy wiadomości) * sentAt: LocalDateTime (data i godzina wysłania wiadomości) * content: String (treść wiadomości; dla wiadomości plikowych może być pusty) * chatId: Long (identyfikator czatu, do którego należy wiadomość) * attachedFiles: List<ChatMessageAttachedFileDto> (lista załączonych plików) – hasNextSlice: Boolean (czy istnieje kolejna „strona” / porcja wiadomości) – numberOfMessages: Integer (liczba wiadomości zwróconych w tej odpowiedzi) – sliceNumber: Integer (numer bieżącej porcji wiadomości) |
|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Tabela 7.46: Karta endpointu: /chats/{chatId}/messages

| KARTA ENDPOINTU API | |
|-----------------------|-----------------------------------|
| Identyfikator: | EP39 |
| Ścieżka: | /chats/get-or-create-private-chat |

| | |
|-----------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Nazwa: | Pobierz istniejący lub utwórz nowy prywatny czat z użytkownikiem |
| Parametry wejściowe: | Brak |
| Query params: | <ul style="list-style-type: none"> • receiverUsername: String (nazwa użytkownika, z którym chcemy rozpocząć lub kontynuować rozmowę) • chatId: Long (opcjonalnie, identyfikator istniejącego czatu – jeśli jest już znany) |
| Kod(y) statusu odpowiedzi: | 200 OK, 401 Unauthorized, 404 Not Found |
| Dane zwracane: | <ul style="list-style-type: none"> • body: ChatDto (szczegóły czatu), zawiera: <ul style="list-style-type: none"> – id: Long (identyfikator czatu) – name: String (nazwa czatu – nazwa grupy lub nazwa rozmówcy) – lastMessage: ChatMessageDto (ostatnia wiadomość w czacie, jeśli istnieje) – imageUrl: String (URL avatara czatu lub rozmówcy) – messages: List<ChatMessageDto> (lista wiadomości zwróconych razem z czatem) – chatType: ChatType (typ czatu: PRIVATE lub GROUP) – participants: List<ChatParticipantDto> (lista uczestników czatu) |

Tabela 7.47: Karta endpointu: /chats/get-or-create-private-chat

| KARTA ENDPOINTU API | |
|-----------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Identyfikator: | EP40 |
| Ścieżka: | /chats/{chatId}/send-files |
| Nazwa: | Wyślij jeden lub wiele plików w ramach czatu |
| Parametry wejściowe: | <ul style="list-style-type: none"> • chatId: Long (identyfikator czatu w ścieżce) • media: List<MultipartFile> (lista załączanych plików do wysłania w wiadomości) |
| Query params: | Brak |
| Kod(y) statusu odpowiedzi: | 201 Created, 401 Unauthorized, 404 Not Found, 413 Payload Too Large, 415 Unsupported Media Type, 500 Internal Server Error |
| Dane zwracane: | <ul style="list-style-type: none"> • Brak (pusta odpowiedź; wiadomości z plikami pojawią się w historii czatu) |

Tabela 7.48: Karta endpointu: /chats/{chatId}/send-files

| KARTA ENDPOINTU API | |
|-----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Identyfikator: | EP41 |
| Ścieżka: | /chats/create/group |
| Nazwa: | Utwórz nowy czat grupowy |
| Parametry wejściowe: | <ul style="list-style-type: none"> • body: CreateGroupChatDto (dane nowego czatu grupowego), zawiera: <ul style="list-style-type: none"> – usermes: List<String> (lista nazw użytkowników, którzy mają zostać uczestnikami czatu) – ownerUsername: String (nazwa właściciela / twórcy czatu) |

| | |
|-----------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Query params: | Brak |
| Kod(y) statusu odpowiedzi: | 201 Created, 400 Bad Request, 401 Unauthorized |
| Dane zwracane: | <ul style="list-style-type: none"> • body: ChatDto (utworzony czat grupowy), zawiera: <ul style="list-style-type: none"> – id: Long (identyfikator czatu) – name: String (nazwa czatu – nazwa grupy) – lastMessage: ChatMessageDto (ostatnia wiadomość w czacie, jeśli istnieje) – imageUrl: String (URL avatara czatu) – messages: List<ChatMessageDto> (lista wiadomości zwróconych razem z czatem) – chatType: ChatType (typ czatu, PRIVATE lub GROUP) – participants: List<ChatParticipantDto> (lista uczestników czatu) |

Tabela 7.49: Karta endpointu: /chats/create/group

| KARTA ENDPOINTU API | |
|-----------------------|---------------------------------------------------|
| Identyfikator: | EP42 |
| Ścieżka: | /chats/{chatId} |
| Nazwa: | Zaktualizuj dane czatu grupowego (nazwa, zdjęcie) |

| | |
|-----------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Parametry wejściowe: | <ul style="list-style-type: none"> • chatId: Long (identyfikator czatu grupowego) • updateGroupChatDto: UpdateGroupChatDto (wysyłany jako multipart/form-data, zawiera dane do zmiany, nowa nazwa, nowe zdjęcie) |
| Query params: | Brak |
| Kod(y) statusu odpowiedzi: | 200 OK, 401 Unauthorized, 404 Not Found, 413 Payload Too Large, 415 Unsupported Media Type, 500 Internal Server Error |
| Dane zwracane: | <ul style="list-style-type: none"> • body: UpdatedGroupChatDto (zaktualizowane dane czatu grupowego), zawiera: <ul style="list-style-type: none"> – newName: String (aktualna nazwa czatu po zmianie) – newImgUrl: String (aktualny URL obrazka grupy po zmianie) |

Tabela 7.50: Karta endpointu: /chats/{chatId}

| KARTA ENDPOINTU API | |
|-----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| Identyfikator: | EP43 |
| Ścieżka: | /chats/group-chat/add/search/{chatId} |
| Nazwa: | Wyszukaj potencjalnych użytkowników do dodania do czatu grupowego |
| Parametry wejściowe: | <ul style="list-style-type: none"> • chatId: Long (identyfikator czatu grupowego, do którego chcemy dodać użytkowników) |

| | |
|-----------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Query params: | <ul style="list-style-type: none"> • query: String (fraza wyszukiwania po nazwie użytkownika) • page: Integer (numer strony wyników, domyślnie 0) • size: Integer (liczba wyników na stronę, domyślnie 20) |
| Kod(y) statusu odpowiedzi: | 200 OK, 401 Unauthorized, 404 Not Found |
| Dane zwracane: | <ul style="list-style-type: none"> • body: SimpleSliceDto<PotentialChatMemberDto> (stworzona lista potencjalnych uczestników czatu), zawiera: <ul style="list-style-type: none"> – hasNext: boolean (czy istnieje kolejna „strona” wyników) – collection: Collection<PotentialChatMemberDto> (kolekcja potencjalnych użytkowników), każdy element: <ul style="list-style-type: none"> * username: String (nazwa użytkownika) * profileImg: String (URL zdjęcia profilowego użytkownika) |

Tabela 7.51: Karta endpointu: /chats/group-chat/add/search/{chatId}

7.1.3 Integracja z bazą danych

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

W systemie zaimplementowano zestaw najistotniejszych tabel, które opisano poniżej:

- **chat-invitations** — przechowuje zaproszenia do czatów wysyłane użytkownikom.
- **chat-message-attached-file** — przechowuje pliki dołączone do wiadomości w czatach.
- **chat-messages** — zapisuje wiadomości wysyłane w czatach.
- **chat-participants** — zawiera informacje o uczestnikach poszczególnych czatów.
- **chats** — lista czatów dostępnych w systemie.
- **favorite-spots** — informacje o miejscach (spotach) oznaczonych jako ulubione przez użytkowników.
- **forum-categories** — kategorie, do których przypisywane są posty na forum.
- **forum-tags** — tagi przypisywane postom na forum.
- **friendships** — relacje znajomości między użytkownikami.
- **media** — ogólne media przesyłane przez użytkowników na forum (zdjęcia, filmy).
- **post-comment-down-votes** — przechowuje „minusy” nadawane komentarzom do postów.
- **post-comment-reports** — raporty zgłasiane przez użytkowników wobec komentarzy.
- **post-comment-up-votes** — przechowuje „plusy” nadawane komentarzom do postów.
- **post-comments** — komentarze użytkowników do postów.
- **post-down-votes** — „minusy” nadawane postom.
- **post-followers** — informacje o użytkownikach obserwujących dany post.

- **post-reports** — raporty zgłasiane wobec postów.
- **post-tags** — tagi przypisane do konkretnych postów.
- **post-up-votes** — „plusy” nadawane postom.
- **posts** — posty tworzone przez użytkowników.
- **spot-comment-down-votes** — „minusy” nadawane komentarzom do spotów.
- **spot-comment-media** — pliki multimedialne dołączone do komentarzy przy spotach.
- **spot-comment-up-votes** — „plusy” nadawane komentarzom do spotów.
- **spot-comments** — komentarze użytkowników do spotów.
- **spot-media** — pliki multimedialne związane z konkretnymi spotami.
- **spots** — baza spotów w systemie.
- **spots-tags** — tagi przypisane do poszczególnych spotów.
- **tags-of-spots** — alternatywna tabela z tagami dla spotów.
- **user-followed-posts** — lista postów śledzonych przez użytkowników.
- **user-followers** — relacje obserwujących użytkowników.
- **user-liked-spot-media** — informacja o polubieniach mediów powiązanych ze spotami.
- **users** — dane użytkowników systemu.

7.1.4 Obsługa uwierzytelnienia

7.1.5 Konteneryzacja

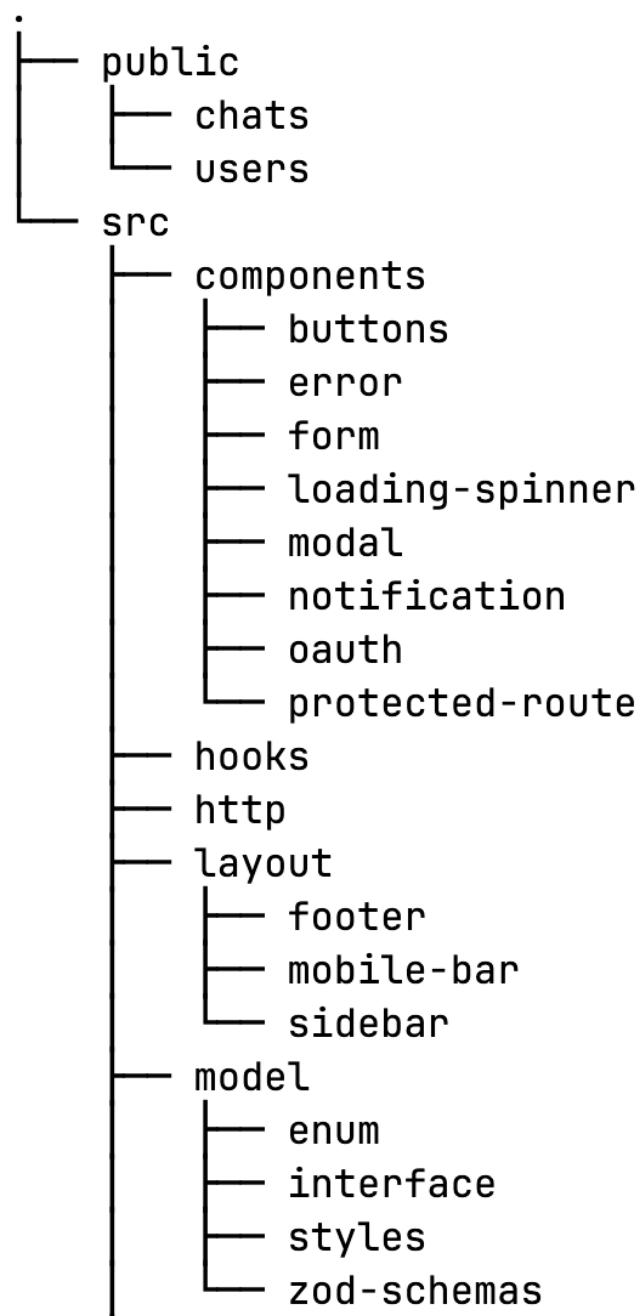
7.2 Implementacja frontendu

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

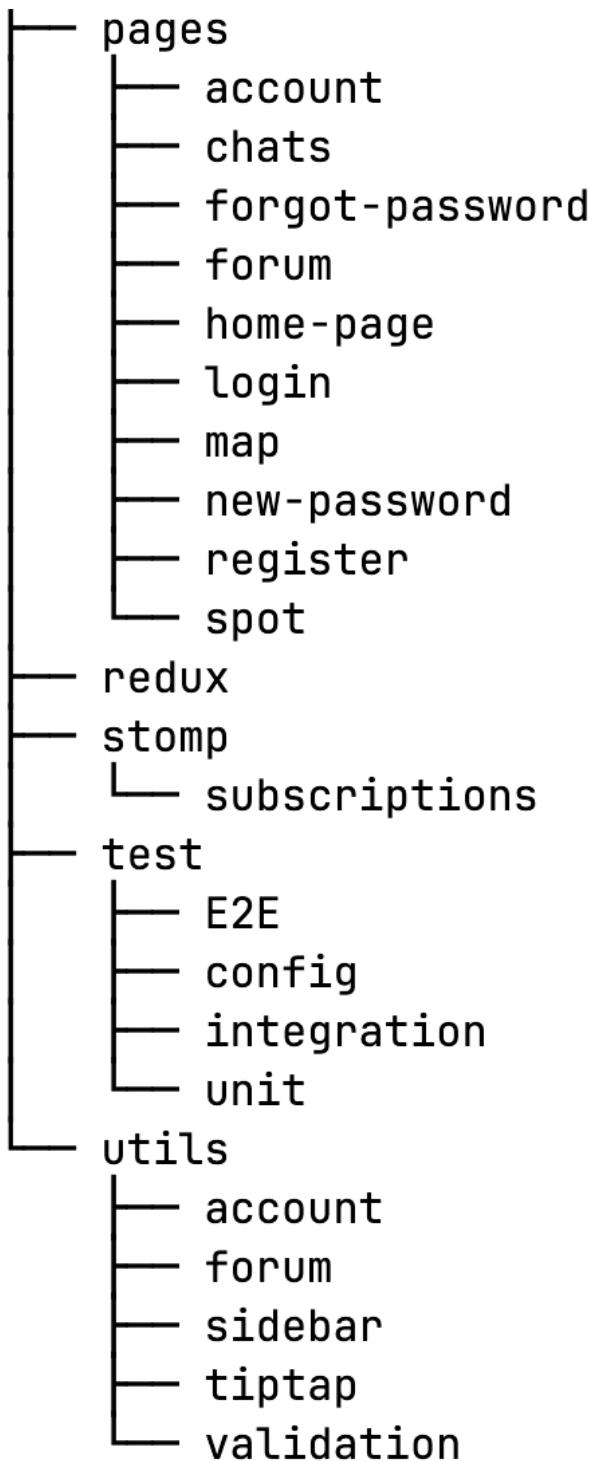
7.2.1 Struktura aplikacji

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

Architekturę aplikacji [frontendowej](#) zaprojektowano w strukturze [Folder by type](#), która polega na podziale kodu według typu zasobu (komponenty, strony, modele itd.). Każdy plik znajduje się w katalogu odpowiadającym jego przeznaczeniu, co przedstawiono na rysunkach [7.3](#) oraz [7.4](#).



Rysunek 7.3: Struktura katalogów (1)



Rysunek 7.4: Struktura katalogów (2)

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

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

Rysunek 7.5: Implementacja routera (1)

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

export default router; Show usages ⚡ Adam Langmesser
```

Rysunek 7.6: Implementacja routera (2)

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

rolę bramki (rysunek 7.7).

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

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

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

Rysunek 7.7: Implementacja komponentu bramki (`ProtectedRoute`)

7.2.2 Zarządzanie stanem i przepływ danych

W niniejszym podrozdziale opisano zastosowane w projekcie podejście do zarządzania **stanem** oraz organizację przepływu danych w aplikacji frontendowej.

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

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

Z racji tego, że korzystamy z **reacta** w połączeniu z **TypeScriptem**, przygoto-

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

```

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

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

```

Rysunek 7.8: Konfiguracja sklepu (Redux store)

```

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

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

export const accountSlice : Slice<AccountSliceProps, { 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 7.9: Przykładowy slice odpowiedzialny za sprawdzenie czy użytkownik jest zalogowany

7.2.3 Integracja i komunikacja z backendem

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

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

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

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

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

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

Rysunek 7.10: Implementacja modułu `account` (1)

```

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

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

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

```

Rysunek 7.11: Implementacja modułu account (2)

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

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

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

Rysunek 7.12: Wykorzystanie TanStack Query przy logowaniu użytkownika

7.2.4 Style

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

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



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

Rysunek 7.13: Implementacja zmiennych kolorystycznych (1)



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

Rysunek 7.14: Implementacja zmiennych kolorystycznych (2)

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

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

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

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

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

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

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

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

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

Rysunek 7.17: Implementacja animacji z wykorzystaniem Motion

7.2.5 Wyszukiwarka spotów

W niniejszym rozdziale przedstawiono sposób implementacji wyszukiwarki spotów.

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

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

Rysunek 7.18: Implementacja prostej wersji wyszukiwarki

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

Rysunek 7.19: Implementacja zaawansowanej wersji wyszukiwarki

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

```
<div className="■dark:shadow-darkBgSoft flex rounded-full shadow-lg shadow-black/20">
  <NavLink
    to="/"
    className={({ isActive } : NavLinkRenderProps) : string =>
      `■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 =>
      `■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 7.20: Implementacja komponentu do przełączania trybów

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

```

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

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

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

```

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

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

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

- `Switch` – służy do przełączania widoku między trybem podstawowym a zaawansowanym,
- `SearchBar` – podstawowa wyszukiwarka `spotów`,
- `Carousel` – wyświetla najpopularniejsze `spotty`,
- `SearchSpotList` – wyświetla wyszukane `spotty`.

W skład zaawansowanej wersji wchodzą następujące komponenty:

- `Switch` – służy do przełączania widoku między trybem podstawowym a zaawansowanym,
- `AdvanceSearchBar` – zaawansowana wyszukiwarka `spotów`,
- `SearchSpotList` – wyświetla wyszukane `spotty`.

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

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

```

<div className="■dark:bg-darkBgSoft □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"; readonly id: string} : 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>
    <button
      className="■dark:bg-darkBgMuted ■dark:hover:bg-darkBgMuted/80 □bg-lightBgMuted
      □hover:bg-lightBgMuted/80 flex w-full cursor-pointer justify-center rounded-md p-2 md:w-fit"
      onClick={handleSearchSpots}
    >
      <FaSearch />
    </button>
  </div>
</div>

```

Rysunek 7.22: Implementacja prostej wyszukiwarki

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

```

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

```

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

Komponent **SpotTile** zawiera następujące informacje:

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

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

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

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

7.4 Zastosowanie WebSocket

Niniejszy rozdział prezentuje implementację protokołu [WebSocket](#) na potrzeby realizacji modułu czatu. Opis przygotowano w oparciu o specyfikację [RFC6455](#) oraz dokumentację [Spring Framework](#) dotyczącą [WebSocket](#) i [STOMP](#) [20, 21, 22, 23].

7.4.1 Charakterystyka protokołu [WebSocket](#)

Na podstawie specyfikacji [RFC6455](#), opracowanej przez [IETF](#), protokół [WebSoc-](#)[ket](#) można opisać jako mechanizm komunikacji pomiędzy aplikacją działającą w przeglądarce a [serwer](#), który umożliwia dwukierunkową wymianę danych w czasie zbliżonym do rzeczywistego. Przeglądarka utrzymuje stałe połączenie z [serwerem](#), dzięki czemu może na bieżąco wysyłać dane, a [serwer](#) w dowolnym momencie przesyłać odpowiedzi, bez konieczności ciągłego odświeżania strony ani inicjowania wielu osobnych zapytań [HTTP](#). Tego typu komunikacja jest możliwa wyłącznie z [serwerami](#), które świadomie ją dopuszczają, co zwiększa kontrolę nad bezpieczeństwem. Przeglądarka w dalszym ciągu stosuje model bezpieczeństwa oparty na pochodzeniu (origin), weryfikując domenę, z której pochodzi kod. Technologia

[WebSocket](#) została zaprojektowana z myślą o aplikacjach webowych, takich jak czaty, powiadomienia w czasie rzeczywistym, aplikacje giełdowe czy gry sieciowe, które wymagają stałego kanału komunikacji z [serwerem](#), zamiast opierać się na wielu tradycyjnych połączeniach [HTTP](#) i okresowym odpytywaniu [serwera](#).

Warto podkreślić, że sam protokół [WebSocket](#) definiuje przede wszystkim **mechanizm utrzymania stałego połączenia i przesyłania ramek danych**. Nie narzuca natomiast **struktury komunikatów aplikacyjnych**, sposobu ich adresowania, routingu ani modelu [subskrypcji](#). Oznacza to, że po ustanowieniu połączenia [WebSocket](#) aplikacja musi samodzielnie ustalić, jak wygląda format wiadomości oraz w jaki sposób realizowane jest publikowanie i odbieranie zdarzeń.

Z tego powodu często stosuje się protokoły warstwy aplikacyjnej działające nad [WebSocket](#), np. [STOMP](#) (*Simple Text Oriented Messaging Protocol*). Zapewnia on ustandaryzowaną semantykę komunikacji w modelu wiadomościowym, w szczególności:

- **adresowanie wiadomości** do tzw. [destynacji](#),
- **mechanizm subskrypcji** ([SUBSCRIBE](#)) oraz dystrybucję wiadomości w stylu [publish–subscribe](#),
- **nagłówki i metadane** wiadomości (np. identyfikatory, typy zdarzeń, dodatkowy kontekst),
- **potwierdzanie odbioru** ([ACK](#)) i lepszą kontrolę nad niezawodnością dostarczenia,
- **obsługę błędów** na poziomie protokołu (ramki [ERROR](#)) oraz opcjonalne **heartbeat** do wykrywania zerwanych połączeń.

7.4.2 Zastosowanie [WebSocket](#) w naszym projekcie

Główna metodą komunikacji między [backendem](#) a [frontendem](#) w naszej aplikacji jest [REST API](#) oparte na protokole [HTTP](#). Tego typu interfejs bardzo dobrze sprawdza się w typowych scenariuszach, w których [klient \(frontend\)](#) inicjuje żądanie, a [serwer \(backend\)](#) zwraca odpowiedź, np. podczas logowania użytkownika,

pobierania listy spotów czy dodawania komentarza na forum. Każda operacja ma wówczas postać pojedynczego, niezależnego wywołania [HTTP](#), po którym połączenie jest zamkane.

Problem pojawia się jednak wtedy, gdy konieczna jest ciągła wymiana danych w czasie zbliżonym do rzeczywistego, a więc nie tylko na żądanie [klienta](#), lecz także z inicjatywy [serwera](#). W naszej aplikacji taka potrzeba wystąpiła przy implementacji modułu czatu. Teoretycznie czat można zrealizować wyłącznie w oparciu o [REST API](#), na przykład poprzez [polling](#): [klient](#) cyklicznie wysyłałby zapytanie o nowe wiadomości, a [serwer](#) zwracałby aktualny stan konwersacji. Można również stosować [long polling](#), w którym żądanie jest utrzymywane dłużej, aż pojawi się nowa wiadomość.

Takie podejście ma jednak istotne wady. Częste odpytywanie [serwera](#) generuje dużą liczbę żądań [HTTP](#), obciążając zarówno [serwer](#), jak i sieć, mimo że w wielu przypadkach odpowiedzi nie zawierają żadnych nowych danych. Dodatkowo wprowadza to opóźnienia: użytkownik zobaczy nową wiadomość dopiero przy kolejnym odpytywaniu, a nie w momencie jej wysłania. Rozwiązania oparte na [long pollingu](#) są z kolei bardziej złożone w implementacji i wciąż nie zapewniają tak naturalnego, dwukierunkowego kanału komunikacji, jakiego oczekuje się od wspólnego czatu.

Z tego powodu wszędzie tam, gdzie potrzebna jest stała, dwukierunkowa komunikacja w czasie rzeczywistym, stosuje się protokół [WebSocket](#). W naszej aplikacji został on wykorzystany właśnie w module czatu, co pozwala na utrzymanie jednego trwałego połączenia pomiędzy [klientem](#) a [serwerem](#) i natychmiastowe dostarczanie wiadomości do wszystkich uczestników konwersacji, bez konieczności ciągłego odpytywania [serwera](#) za pomocą [REST API](#).

7.4.3 Implementacja na backendzie

[Spring Framework](#) udostępnia dwa popularne podejścia do obsługi komunikacji [WebSocket](#). Pierwsze to praca na „surowych” [ramkach WebSocket](#) (pełna kontrola, ale konieczność zaprojektowania własnego formatu komunikatów i dystrybucji). Drugie wykorzystuje protokół [STOMP](#) jako warstwę aplikacyjną nad [WebSocket](#).

W niniejszym projekcie wybrano wariant z **STOMP**, ponieważ upraszcza on implementację czatu: pozwala korzystać z gotowych **destynacji** (ścieżek), **subskrypcji** oraz mechanizmu publikowania wiadomości.

7.4.3.1 Plik `WebSocketConfig.java` — konfiguracja **WebSocket**

Na rysunku 7.24 przedstawiono konfigurację **WebSocket**. Definiuje ona endpoint do zestawienia połączenia oraz dwa **prefiksy destynacji**: dla wiadomości przychodzących do aplikacji oraz dla wiadomości rozsyłanych do **klientów**.

```

@Configuration  Adam Langmesser
@EnableWebSocketMessageBroker
public class WebSocketConfig implements WebSocketMessageBrokerConfigurer {

    @Override  1 usage  Adam Langmesser
    public void configureMessageBroker(MessageBrokerRegistry config) {
        config.enableSimpleBroker( ...destinationPrefixes: "/subscribe");
        config.setApplicationDestinationPrefixes("/app");
    }

    @Override  1 usage  Adam Langmesser
    public void registerStompEndpoints(StompEndpointRegistry registry) {
        registry.addEndpoint( ...paths: "/connect")
            .setAllowedOriginPatterns("http://localhost:*")
            .withSockJS();
    }

    @Override  1 usage  Adam Langmesser
    public void configureClientInboundChannel(ChannelRegistration registration) {
        registration.interceptors(securityContextChannelInterceptor());
    }

    @Bean  Adam Langmesser
    public SecurityContextChannelInterceptor securityContextChannelInterceptor() {
        return new SecurityContextChannelInterceptor();
    }
}

```

Rysunek 7.24: Konfiguracja WebSocket w module czatu (plik `WebSocketConfig.java`).

Najważniejsze elementy konfiguracji:

- `/connect` – endpoint zestawienia połączenia (tzw. `handshake`); dodatkowo włączono `SockJS`.
- `/app/*` – prefiks destynacji dla wiadomości wysyłanych **od klienta do serwera** (obsługiwanych przez metody z `@MessageMapping`).
- `/subscribe/*` – prefiks destynacji dla wiadomości wysyłanych **od serwera**

do klientów (klient subskrybuje te kanały, a **serwer** publikuje tam komunikaty).

- **SecurityContextChannelInterceptor** – zapewnia dostęp do kontekstu bezpieczeństwa dla wiadomości przychodzących (backend rozpoznaje, kto wysłał wiadomość).

7.4.3.2 Plik ChatStompCommunicationController.java — kontroler STOMP

Rysunek 7.25 pokazuje kontroler STOMP, który jest punktem wejścia dla nowych wiadomości wysyłanych przez użytkownika.

```
@Slf4j  Adam Langmesser
@Controller
@RequiredArgsConstructor
public class ChatStompCommunicationController {

    private final ChatStompCommunicationService chatStompCommunicationService;
    private final ChatService chatService;

    @MessageMapping("/send/{chatId}/message")  Adam Langmesser
    public void sendChatMessage(@DestinationVariable String chatId, @Payload IncomingChatMessageDto message) {
        log.debug("Received message for chat: {}, message: {}", chatId, message);
        var chatMessageDtoToBroadcast = chatService.saveChatMessage(message);
        chatStompCommunicationService.broadcastChatMessageToAllChatParticipants(chatMessageDtoToBroadcast);
        chatStompCommunicationService.broadcastACKVersionToSender(chatMessageDtoToBroadcast, message.optimisticMessageUUID());
    }
}
```

Rysunek 7.25: Kontroler STOMP odbierający wiadomości czatu (plik ChatStompCommunicationController.java).

Kontroler udostępnia wejściową **destynację**: /app/send/{chatId}/message. Jej rola jest następująca:

1. odebrać wiadomość od **klienta**,
2. zlecić zapis wiadomości do **Baza danych** (serwis domenowy czatu),
3. uruchomić rozesłanie wiadomości,
4. oddelegować wysłanie potwierdzenia (**ACK**) do nadawcy wiadomości.

7.4.3.3 Plik ChatStompCommunicationService.java — serwis dystrybucji wiadomości

Rysunek 7.26 przedstawia serwis, który odpowiada za publikowanie komunikatów na kanały `subskrypcji /subscribe/*`. Serwis ten jest wykorzystywany zarówno przez kontroler STOMP, jak i przez inne elementy logiki modułu czatu.

```
@Slf4j 3 usages  Adam Langmesser
@Service
@RequiredArgsConstructor
public class ChatStompCommunicationService {

    private final SimpMessagingTemplate messagingTemplate;
    private final ChatRepository chatRepository;
    private final CustomUserDetailsService customUserDetailsService;

    Langmesser, 03/07/2025 22:15 · Feature/mer 243 (#123)

    Broadcasts a chat message to all participants of the chat.
    The STOMP endpoint for this method is dynamically created based on the username of each
    chat participant. This allows each participant to receive messages sent to the chat they are part
    of.
    Each User should listen on its own channel: /subscribe/chats/{username}.
    Each User has one universal channel to receive new messages in real time for all chats they are
    part of.

    Params: chatMessageDto - the chat message to be broadcasted

    @Transactional 2 usages  Adam Langmesser
    public void broadcastChatMessageToAllChatParticipants(ChatMessageDto chatMessageDto) {
        var chatParticipants = chatRepository.findById(chatMessageDto.chatId())
            .orElseThrow(() -> new IllegalArgumentException("Chat not found"))
            .getParticipants();
        log.info("Broadcasting chat message to {} participants: {}", (long) chatParticipants.size(), chatMessageDto);
        chatParticipants.forEach(ChatParticipant participant -> messagingTemplate.convertAndSend(
            destination: "/subscribe/chats/" + participant.getUser().getUsername(), chatMessageDto));
    }

    @Transactional 1 usage  Adam Langmesser
    public void broadcastACKVersionToSender(ChatMessageDto chatMessageDto, String optimisticMessageUUID) {
        var currentUserUsername = customUserDetailsService.loadUserDetailsFromSecurityContext().getUsername();

        log.info("Broadcasting ack for message with id: {} for chat id: {} to {}", chatMessageDto.id(), chatMessageDto.chatId(), currentUserUsername);
        var ackChatMessageDto = new ChatMessageAckDto(
            chatMessageDto, optimisticMessageUUID
        );
        messagingTemplate.convertAndSend(
            destination: "/subscribe/chats/ack/" + currentUserUsername, ackChatMessageDto);
    }
}
```

Rysunek 7.26: Serwis rozsyłający wiadomości czatu i potwierdzenia ACK (plik ChatStompCommunicationService.java).

Serwis realizuje dwa typy wysyłki:

- **Broadcast wiadomości do uczestników czatu** – po zapisaniu wiadomości `serwer` pobiera listę uczestników czatu i wysyła `ChatMessageDto` do każdego z nich na jego kanał użytkownika.
- **ACK do nadawcy** – `serwer` wysyła potwierdzenie `ChatMessageAckDto` na osobny kanał nadawcy, aby `frontend` mógł powiązać wiadomość zapisaną w `Baza danych` z odpowiadającą jej wersją „optymistyczną”.

7.4.3.4 Zestawienie wykorzystywanych destynacji

Poniżej zestawiono wszystkie destynacje używane w komunikacji WebSocket w module czatu:

/connect endpoint zestawienia połączenia WebSocket (tzw. handshake).

/app/send/{chatId}/message Wejście do backendu: klient wysyła nową wiadomość do czatu o identyfikatorze chatId.

/subscribe/chats/{username} Kanał użytkownika: klient subskrybuje i odbiera nowe wiadomości (ze wszystkich czatów, w których uczestniczy).

/subscribe/chats/ack/{username} Kanał potwierdzeń (ACK): klient (nadawca) odbiera potwierdzenie zapisania wiadomości.

Dodatkowo stosowane są dwa prefiksy porządkujące komunikację:

- /app/* – wiadomości od klienta do aplikacji,
- /subscribe/* – wiadomości od aplikacji do klientów (kanały subskrypcji).

7.4.4 Implementacja na frontendzie

W frontendzie zastosowano podejście oparte na jednej współdzielonej instancji serwisu komunikacji, udostępnianej poprzez Kontekst (React). Rozwiązanie składa się z:

- serwisu komunikacyjnego WebSocketService (utrzymanie połączenia, subskrypcje, publikacja),
- hooka useWebSocket (ergonomiczne użycie w komponentach),
- providera WebSocketProvider (utrzymanie cyklu życia połączenia zależnie od logowania),
- fabryk subskrypcji (np. createChatSubscription) aktualizujących Stan w Redux.

7.4.4.1 Plik WebSocketService.ts — klasa WebSocketService

```
Zarządza jednym połączeniem STOMP/WebSocket:  
• inicjalizacja i automatyczne reconnect  
• jednoczesne subskrypcje (wznowione po reconnect)  
• publikacja wiadomości  
  
✓ export class WebSocketService { Show usages Adam Langmesser  
    private client: Client;  
    private subscriptions : Map<string, StompSubscription> = new Map<string, StompSubscription>();  
  
    Przechowuje subskrypcje oczekujące na moment, kiedy połączenie WebSocket/STOMP stanie  
    się aktywne. Prosty mechanizm buforowania  
  
    private pendingSubs : Map<string, (msg: IMessage) => void> = new Map<string, (msg: IMessage) => void>();  
    private isConnected : boolean = false;  
  
Zarządza jednym połączeniem STOMP/WebSocket:  
• inicjalizacja i automatyczne reconnect  
• jednoczesne subskrypcje (wznowione po reconnect)  
• publikacja wiadomości  
  
Params: brokerURL – adres SockJS endpointu, np. "http://localhost:8080/ws"  
  
constructor(private brokerURL: string) { Show usages Adam Langmesser  
    this.client = new Client({  
        webSocketFactory: () : WebSocket => new SockJS(this.brokerURL),  
        reconnectDelay: 5000,  
        debug: (msg: string) : void => console.debug(message: "[STOMP]", msg),  
        onStompError: this.onError.bind(this),  
        onWebSocketClose: this.onClose.bind(this),  
    });  
  
    this.client.onConnect = this.onConnect.bind(this);  
}
```

Rysunek 7.27: Klasa `WebSocketService` w [TypeScript](#) (część 1/4).

```

Jeśli połączenie jeszcze nie zostało nawiązane, to wywołanie connect() je utworzy.

Jeśli połączenie już istnieje (czyli metoda została już wcześniej wywołana i klient jest „aktywny”), to kolejne wywołanie connect() nie spowoduje błędu ani podwójnego połączenia — po prostu nic się nie zmieni.

connect(): void { Show usages  Adam Langmesser
  if (!this.client.active) {
    this.client.activate();
  }
}

Rozłącza i czyści wszystkie subskrypcje.

disconnect(): void { Show usages  Adam Langmesser
  this.client.deactivate();
  this.cleanupAll();
  this.isConnected = false;
}

Internal: Gdy STOMP się połączy – wznowienie pendingSubs.

private onConnect(): void { Show usages  Adam Langmesser
  this.isConnected = true;
  this.pendingSubs.forEach((cb : (msg: IMessage) => void , dest : string ) : void  => {
    const sub : StompSubscription  = this.client.subscribe(dest, cb);
    this.subscriptions.set(dest, sub);
  });
  this.pendingSubs.clear();
}

Internal: Obsługa niespodziewanego zamknięcia socketu.

private onClose(evt: CloseEvent): void { Show usages  Adam Langmesser
  console.warn( message: "WebSocket closed:", evt.reason);
  this.isConnected = false;
  // stomp.js sam spróbuje reconnect
}

```

Rysunek 7.28: Klasa `WebSocketService` w [TypeScript](#) (część 2/4).

```

Internal: Obsługa błędów STOMP frame z brokerem.

private onError(frame: Frame): void { Show usages & Adam Langmesser
    console.error( message: "STOMP error:", frame.headers["message"], frame.body);
}

Subskrybuje dany topic.

Params: destination – np. "/topic/chat"
        callback – wywoływany na każdy komunikat

subscribe(destination: string, callback: (msg: IMessage) => void): void { Show usages
    // odsubskrybij poprzednią, jeśli była
    if (this.subscriptions.has(destination)) {
        this.subscriptions.get(destination)!.unsubscribe();
        this.subscriptions.delete(destination);
    }
    if (this.isConnected) {
        const sub :StompSubscription = this.client.subscribe(destination, callback);
        this.subscriptions.set(destination, sub);
    } else {
        // zapisz do wznowienia po connect
        this.pendingSubs.set(destination, callback);
    }
}

Usuwa subskrypcję z danego topicu.

Params: destination – identyfikator subskrypcji

unsubscribe(destination: string): void { Show usages & Adam Langmesser
    if (this.subscriptions.has(destination)) {
        this.subscriptions.get(destination)!.unsubscribe();
        this.subscriptions.delete(destination);
    }
    this.pendingSubs.delete(destination);
}

```

Rysunek 7.29: Klasa `WebSocketService` w [TypeScript](#) (część 3/4).

Publikuje wiadomość JSON-ując payload.

Params: destination – np. "/app/chat.send"
 payload – dowolny obiekt
 headers – dodatkowe nagłówki STOMP

```
publish( Show usages & Adam Langmesser
  destination: string,
  payload: any,
  headers: Record<string, string> = {},
): void {
  if (!this.isConnected) {
    console.warn(
      message: `STOMP not connected, cannot publish to ${destination}`,
    );
    return;
  }
  this.client.publish({
    destination,
    body: JSON.stringify(payload),
    headers,
  });
}
```

Czy klient jest aktualnie połączony?

```
get connected(): boolean { Show usages & Adam Langmesser
  return this.isConnected;
}
```

Internal: Pełne sprzątanie przy disconnect.

```
private cleanupAll(): void { Show usages & Adam Langmesser
  this.subscriptions.forEach((sub: StompSubscription) : void => sub.unsubscribe());
  this.subscriptions.clear();
  this.pendingSubs.clear();
}
```

Rysunek 7.30: Klasa `WebSocketService` w [TypeScript](#) (część 4/4).

Klasa `WebSocketService` stanowi warstwę po stronie [frontendu](#), która zarządza jednym połączeniem [STOMP](#) działającym nad [WebSocket](#). Do zestawienia transportu wykorzystano [SockJS](#), a właściwy protokół wiadomościowy realizuje [klient](#)

STOMP z biblioteki [StompJS](#).

Główne odpowiedzialności klasy:

- inicjalizacja i utrzymanie połączenia ([handshake](#) oraz automatyczny [Reconnect](#)),
- rejestrowanie i usuwanie [subskrypcji STOMP](#),
- buforowanie subskrypcji, które zostały zadeklarowane przed nawiązaniem połączenia,
- publikowanie wiadomości do wskazanej [destynacji](#) oraz dołączanie [nagłówków](#).

Pola klasy:

- `client` – instancja [klient STOMP](#), która realizuje połączenie i wymianę wiadomości.
- `subscriptions` – mapa aktywnych subskrypcji (klucz: [destynacja](#), wartość: uchwyt subskrypcji).
- `pendingSubs` – bufor oczekujących subskrypcji: `destynacja` → [Callback](#).
- `isConnected` – flaga stanu połączenia.

Konstruktor `constructor(brokerURL):`

- konfiguruje `WebSocketFactory` tak, aby transport był realizowany przez [SockJS](#),
- ustawia opóźnienie automatycznego [Reconnect](#) (w kodzie: `reconnectDelay = 5000 ms`),
- podpina obsługę zdarzeń: poprawne połączenie, błąd protokołu oraz zamknięcie połączenia.

`connect()`: uruchamia połączenie wywołując `activate()` na kliencie STOMP. Jeżeli klient jest już aktywny, metoda nie powoduje utworzenia drugiego połączenia.

`disconnect()`: rozłącza klienta (`deactivate()`), usuwa wszystkie subskrypcje (`cleanup`) oraz ustawia `isConnected = false`. Zapobiega to pozostawieniu nieużywanych kanałów po stronie `frontendu`.

`onConnect()`: jest wywoływana po udanym połączeniu i odpowiada za „odtworzenie” subskrypcji: zawartość `pendingSubs` zostaje zamieniona na aktywne subskrypcje w `subscriptions`, po czym bufor jest czyszczony.

`subscribe(destination, callback)`: zakłada subskrypcję na danej `destynacji`. Jeżeli subskrypcja o tej samej destynacji już istnieje, jest usuwana (uniknięcie wielokrotnego odbioru). Gdy połączenie nie jest aktywne, subskrypcja trafia do bufora `pendingSubs` i zostanie zarejestrowana po połączeniu.

`publish(destination, payload, headers)`: publikuje wiadomość tylko wtedy, gdy połączenie jest aktywne. Dane `payload` są poddawane `Serializacja` do formatu `JSON`, a następnie wysyłane do brokera. Opcjonalnie dołączane są `nagłówki` protokołu STOMP.

7.4.4.2 Plik useWebSocket.ts — hook useWebSocket

```
Publikuje wiadomość JSON-ując payload.

Params: destination – np. "/app/chat.send"
        payload – dowolny obiekt
        headers – dodatkowe nagłówki STOMP

publish( Show usages  Adam Langmesser
  destination: string,
  payload: any,
  headers: Record<string, string> = {},
): void {
  if (!this.isConnected) {
    console.warn(
      message: `STOMP not connected, cannot publish to ${destination}`,
    );
    return;
  }
  this.client.publish({
    destination,
    body: JSON.stringify(payload),
    headers,
  });
}

Czy klient jest aktualnie połączony?

get connected(): boolean { Show usages  Adam Langmesser
  return this.isConnected;
}

Internal: Pełne sprzątanie przy disconnect.

private cleanupAll(): void { Show usages  Adam Langmesser
  this.subscriptions.forEach((sub: StompSubscription) : void => sub.unsubscribe());
  this.subscriptions.clear();
  this.pendingSubs.clear();
}
}
```

Rysunek 7.31: Hook useWebSocket w TypeScript (część 1/2).

```

    /**
     * Wysyła wiadomość do brokeru.
     */
    function publish( Show usages  Adam Langmesser
      destination: string,
      payload: any,
      headers?: Record<string, string>,
    ) : void {
      ws.publish(destination, payload, headers);
    }

    /** Status połączenia */
    const connected :boolean = ws.connected;

    return { publish, connected };
}

```

Rysunek 7.32: Hook `useWebSocket` w TypeScript (część 2/2).

Hook `useWebSocket` upraszcza korzystanie z komunikacji [WebSocket/STOMP](#) w [React](#). Zapewnia automatyczne zakładanie i zdejmowanie [subskrypcji](#) zgodnie z cykl [życia komponentu](#) komponentu oraz udostępnia funkcję do wysyłania wiadomości.

Typ `SubscriptionDef`:

- `destination` – [destynacja](#) protokołu [STOMP](#),
- `callback` – [Callback](#) uruchamiany dla każdej odebranej wiadomości.

Działanie `useWebSocket(options)`:

- pobiera instancję serwisu z `useWebSocketService()` (z [Kontekst \(React\)](#)),
- w `useEffect` zakłada subskrypcje podczas [montowanie](#) komponentu oraz usuwa je podczas [odmontowanie](#) (lub przy zmianie zależności).

Ważna uwaga o `tablica zależności`: w implementacji śledzone są tylko same `destination`. Oznacza to, że zmiana samej funkcji `callback` bez zmiany destynacji nie wymusi ponownego zasubskrybowania (jest to zwykle akceptowalne, ale należy o tym pamiętać).

7.4.4.3 Plik WebSocketContext.tsx — Kontekst (React) i provider połączenia

```
const WS_URL :string = process.env.REACT_APP_WS_URL || "http://localhost:8080/connect";
const wsService = new WebSocketService(WS_URL);
const WebSocketContext :Context<WebSocketService> = createContext<WebSocketService>(wsService);
export const WebSocketProvider: React.FC<{ children: React.ReactNode }> = ({ children, Show usages Adam Langmesser
  ...
}, {children: ReactNode} ) :Element => {
  const { isLoggedIn, username } = useSelectorTyped(
    | state: RootState ) :AccountSliceProps => state.account,
  );
  const selectedChatId :number | null = useSelectorTyped( s :(account: AccountSliceProps; notification... ) :number | null => s.chats.selectedChatId);
  const dispatch :ThunkDispatch<account: AccountSliceProp... = useDispatchTyped();
  ...
  const selectedRef :MutableRefObject<number | null> = useRef<number | null>( initialValue: null );
  useEffect(() :void => {
    selectedRef.current = selectedChatId;
  }, [selectedChatId]);
  ...
  useEffect(() :void => {
    if (isLoggedIn) wsService.connect();
    else wsService.disconnect();
  }, [isLoggedIn]);
  ...
  useEffect(() :() => void | undefined => {
    if (!isLoggedIn || !username) return;
    const getSelectedChatId :() => number | null = () :number | null => store.getState().chats.selectedChatId; Show usages Adam Langmesser
    const allSubs: SubscriptionDef[] = [
      createChatSubscription(username, dispatch, getSelectedChatId),
    ];
    const cleanups :() => void[] = allSubs.map((sub :SubscriptionDef ) :() => void => {
      wsService.subscribe(sub.destination, sub.callback);
      return () :void => wsService.unsubscribe(sub.destination);
    });
    return () :void => cleanups.forEach((fn :() => void ) :void => fn());
  }, [isLoggedIn, username, dispatch]);
  ...
  return (
    <WebSocketContext.Provider value={wsService}>
      {children}
    </WebSocketContext.Provider>
  );
};

export const useWebSocketService :() => WebSocketService = () : WebSocketService => Show usages Adam Langmesser
  useContext(WebSocketContext);
```

Rysunek 7.33: Kontekst WebSocketContext oraz komponent WebSocketProvider.

Plik definiuje **Kontekst (React)** React, który udostępnia w całej aplikacji jedną wspólną instancję `WebSocketService`. Dzięki temu wiele komponentów może korzystać z jednego połączenia `WebSocket/STOMP`, zamiast tworzyć osobne połączenia niezależnie.

Adres endpointu: `WS_URL` jest pobierany ze zmiennej środowiskowej `REACT_APP_WS_URL`,

a w razie braku ustawienia przyjmuje wartość domyślną `http://localhost:8080/connect`.

Współdzielona instancja serwisu: `wsService` jest tworzony raz (wzorzec [Singleton](#)) poza komponentem providera. Zapewnia to, że aplikacja utrzymuje jedno połączenie, a nie nowe połączenie przy każdym renderowaniu.

Zarządzanie połączeniem w WebSocketProvider:

- po zmianie `isLoggedIn` provider łączy się (`connect`) lub rozłącza (`disconnect`),
- po zalogowaniu rejestrowane są subskrypcje zależne od użytkownika (np. kanał czatów),
- podczas sprzątania wykonywany jest `cleanup` subskrypcji, aby uniknąć „wyszących” kanałów.

Dodatkowo używany jest [dispatch \(Redux\)](#) (funkcja `dispatch`), aby aktualizować [Stan](#) aplikacji w [Redux](#) w reakcji na wiadomości przychodzące w czasie rzeczywistym.

7.4.4.4 Plik ChatSubscription.ts — fabryka subskrypcji czatu

```
export function createChatSubscription( Show usages & Adam Langmesser
  username: string,
  dispatch: AppDispatch,
  getSelectedChatId: () => number | null,
): SubscriptionDef {
  return {
    destination: `/subscribe/chats/${username}`,
    callback: (msg: IMessage) : void => {
      const payload = JSON.parse(msg.body) as ChatMessageDto;

      dispatch(
        chatActions.setLastMessage({
          chatId: payload.chatId,
          message: payload,
        }),
      );

      const selected : number | null = getSelectedChatId();
      if (selected !== payload.chatId) {
        dispatch(chatActions.markNew(payload.chatId));
      } else {
        dispatch(chatActions.clearNew(payload.chatId));
      }
    },
  };
}
```

Rysunek 7.34: Funkcja `createChatSubscription` tworząca definicję subskrypcji czatu.

Plik definiuje funkcję fabrykującą obiekt typu `SubscriptionDef`, czyli pojedynczą subskrypcję STOMP dla kanału wiadomości czatu danego użytkownika. Jej celem jest odbieranie nowych wiadomości w czasie rzeczywistym oraz aktualizacja Stanu aplikacji w Redux.

Funkcja `createChatSubscription(username, dispatch, getSelectedChatId):`

- buduje destynację subskrypcji: `/subscribe/chats/{username}`,

- w **Callback** wykonuje **Deserializacja** wiadomości z **JSON** (**JSON.parse**),
- wysyła **akcja Redux** do **Redux** w celu aktualizacji „ostatniej wiadomości” w danym czacie,
- porównuje **payload.chatId** z aktualnie wybranym czatem i odpowiednio oznacza czat jako „nowy” lub czyści to oznaczenie.

Zastosowanie funkcji **getSelectedChatId** (zamiast wartości przechowywanej w komponentach) wynika z asynchronicznego charakteru odbioru wiadomości: callback wykonuje się w chwili nadejścia komunikatu, dlatego odczyt bieżącego stanu bezpośrednio ze **Store (Redux)** minimalizuje ryzyko użycia nieaktualnych danych (np. przy szybkim przełączaniu czatów przez użytkownika).

7.4.5 Przebieg komunikacji

[diagram]

1. odniesienia do slwoniczka 2. odniesienia do bibliografii

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 rozpoczęłem 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 scrolllem
- 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

ACK

Skrót od *Acknowledgement*; komunikat potwierdzający odbiór lub przetworzenie wiadomości. W [STOMP](#) może służyć do potwierdzania wiadomości po stronie klienta.. [181](#), [185–187](#)

akcja Redux

Obiekt opisujący zdarzenie/operację zmiany stanu w Redux, obsługiwany przez reducery (np. `chatActions.setLastMessage(...)`). [199](#)

API

(ang. *application programming interface*); zbiór reguł i operacji do komunikacji z oprogramowaniem.. [26](#), [27](#)

Backend

Część aplikacji odpowiedzialna za logikę biznesową, przetwarzanie danych i komunikację z bazą danych. Działa po stronie serwera i obsługuje żądania wysyłane przez frontend. [2](#), [15](#), [24](#), [100](#), [101](#), [103](#), [106](#), [155](#), [166](#), [181](#), [203](#), [216](#)

Backlog

Lista zadań, które należy wykonać w ramach projektu, używane w metodykach zwinnych.. [25](#)

Baza danych

Zbiór uporządkowanych danych przechowywanych w sposób umożliwiający ich łatwe wyszukiwanie, modyfikowanie i analizowanie. W aplikacjach najczęściej wykorzystywane są relacyjne lub nierelacyjne bazy danych. [100](#), [101](#), [103](#), [155](#), [185](#), [186](#)

Biblioteka

Zewnętrzny lub wewnętrzny zestaw gotowych funkcji, klas, komponentów lub modułów, który można wielokrotnie wykorzystywać w projekcie zamiast pisać wszystko od zera. [19](#), [101](#), [155](#), [161](#), [163](#), [166](#), [172](#), [177](#)

BPMN

(ang. *Business Process Model and Notation*); standardowa notacja graficzna, która umożliwia szczegółowe przedstawienie i dokumentowanie procesów biznesowych..
[26](#)

Cache

Mechanizm przechowywania danych w celu przyspieszenia ich ponownego odczytu.
[23](#), [24](#), [100](#), [101](#), [214](#)

Callback

Funkcja przekazywana jako parametr, wywoływana automatycznie przy wystąpieniu zdarzenia (np. gdy nadaje się wiadomość z WebSocket).. [192](#), [195](#), [199](#)

CI/CD

Skrót od *Continuous Integration/Continuous Deployment*. Praktyka programistyczna polegająca na automatyzacji procesu budowania, testowania i wdrażania oprogramowania. [25](#), [202](#), [205](#)

cleanup

„Sprzątanie” zasobów: usuwanie subskrypcji, rozłączanie połączeń i czyszczenie struktur pomocniczych, aby nie pozostawiać aktywnych uchwytów. [193](#), [197](#)

Convention Over Configuration

Zasada programowania polegająca na przyjmowaniu domyślnych, bazowych regów, zamiast ręcznego implementowania konfiguracji. [16](#)

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.. [172](#)

cykl życia komponentu

Etapy działania komponentu (np. montowanie, aktualizacja, odmontowanie) determinujące momenty inicjalizacji i zwalniania zasobów. [195](#)

Deserializacja

Proces odtworzenia obiektu z formatu przenośnego (np. JSON) po stronie odbiorcy. [199](#)

Design

Etap lub proces projektowania wyglądu i funkcjonalności aplikacji, obejmujący zarówno aspekty wizualne, jak i użytkowe (UX/UI). [202](#)

destynacja

Adres logiczny (ścieżka) w komunikacji **STOMP**, na który wysyła się wiadomości lub który się **subskrybuje** (np. /app/... albo /subscribe/...). [181](#), [183–185](#), [187](#), [192](#), [193](#), [195](#), [198](#), [217](#)

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 ciągłe doprecyzowywanie

wymagań i backlogu, oparte na regularnym feedbacku udziałowców. Całość opiera się na praktykach Lean oraz lekkim governance: code review i regularnych przeglądach postępów. . [11](#)

dispatch (Redux)

Funkcja w Redux służąca do wysyłania akcji, które modyfikują stan przechowywany w store. [197](#)

DOM

(ang. *Document Object Model*); interfejs reprezentacji stron internetowych jako węzły i obiekty. Dzięki temu skrypty, np. napisane w JavaScript, mogą wchodzić w interakcje ze stroną (np. modyfikować strukturę, treść lub styl). [19](#)

Droniarz

Potoczne określenie osoby, która jest jednocześnie pilotem oraz operatorem drona. Zwykle entuzjasta dronów.. [9](#), [10](#), [219](#)

Droniarz foto/video

Pilot wykorzystujący drony fotograficzne/filmowe do rejestracji materiałów wizualnych (zdjęcia, wideo), zwykle z naciskiem na stabilizację i jakość obrazu.. [27](#)

endpoint

„Punkt wejścia” komunikacji w aplikacji: konkretny adres/ścieżka, pod którą zestawia się połączenie lub wysyła wiadomości (np. endpoint `handshake /connect`).. [183](#), [184](#), [187](#)

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. [106](#), [158](#)

Framework

Zestaw narzędzi, bibliotek i struktur wspomagających tworzenie aplikacji. Ułatwia programowanie poprzez dostarczenie gotowych komponentów oraz określenie zasad organizacji kodu. [2](#), [15](#), [16](#), [101](#), [106](#), [169](#)

Frontend

Warstwa aplikacji odpowiedzialna za interfejs użytkownika oraz interakcję z użytkownikiem. Zazwyczaj tworzona przy użyciu technologii takich jak HTML, CSS i JavaScript. [2](#), [15](#), [19](#), [24](#), [100](#), [101](#), [103](#), [158](#), [166](#), [181](#), [186](#), [187](#), [191](#), [193](#), [203](#), [211](#)

handshake

Etap początkowy zestawiania połączenia (np. [WebSocket](#)), w którym [klient](#) i [serwer](#) uzgadniają parametry komunikacji.. [184](#), [187](#), [192](#), [210](#)

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`).. [163](#), [164](#), [168](#)

HTTP

Podstawowy protokół komunikacji w sieci WWW, wykorzystywany m.in. przez [REST API](#). Zwykle działa w modelu „żądanie–odpowiedź”, bez stałego połączenia.. [180](#)–[182](#), [213](#)

IDE

(ang. *integrated development environment*); to zintegrowane środowisko programistyczne, służące do tworzenia, modyfikowania, testowania i konserwacji oprogramowania. [24](#)

IETF

Organizacja opracowująca i publikująca standardy internetowe, m.in. dokumenty typu [RFC](#).. [180](#)

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. [178](#)

JSON

Format tekstowy do wymiany danych (JavaScript Object Notation), często używany jako reprezentacja komunikatów API i wiadomości w aplikacjach webowych. [193](#), [199](#)

JVM

(ang. *Java Virtual Machine*); maszyna wirtualna oraz środowisko do wykonywania kodu bajtowego Javy. [16](#)

JWT

Skrót od *JSON Web Token*. Standard służący do bezpiecznego przekazywania informacji między stronami w formacie JSON, często używany w procesach autoryzacji użytkowników. [167](#), [202](#)

klient

Część systemu (zwykle po stronie [Frontendu](#)), która korzysta z usług [serwera](#) i prezentuje dane użytkownikowi.. [181–185](#), [187](#), [207](#), [210](#), [212](#), [213](#), [216](#), [218](#)

klient STOMP

Biblioteka/komponent po stronie klienta realizujący protokół STOMP (subskrypcje, publikacja, ramki, nagłówki) nad transportem WebSocket/SockJS. [191](#), [192](#)

Kontekst (React)

Mechanizm w React pozwalający udostępnić wspólny obiekt (np. serwis WebSocket) wielu komponentom bez przekazywania go przez propsy.. [187](#), [195](#), [196](#)

long polling

Odmiana [pollingu](#): [klient](#) wysyła żądanie, a [serwer](#) wstrzymuje odpowiedź do momentu pojawienia się nowych danych (lub upłynięcia limitu czasu).. [182](#)

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*).. [172](#), [214](#)

montowanie

Moment „podpięcia” komponentu do drzewa UI (pierwsze wyrenderowanie i inicjalizacja). [195](#)

MoSCoW

Metoda nadawania priorytetów wymaganiom, wyróżniająca kategorie: *Must have*, *Should have*, *Could have* oraz *Won't have this time*. [31](#)

Nagłówek

Metadane dołączane do wiadomości lub żądania (np. dodatkowe pola opisujące typ komunikatu, identyfikator, autoryzacje).. [192](#), [193](#)

odmontowanie

Usunięcie komponentu z drzewa UI (moment zwalniania zasobów i zdejmowania subskrypcji). [195](#)

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. . [32](#), [51](#), [52](#), [220](#)

Parametry zapytania (query params)

Pary **klucz=wartość** przekazywane w części adresu URL po znaku zapytania ?, służące m.in. do filtrowania, sortowania, paginacji wyników lub przekazywania dodatkowych opcji żądania do serwera, np. ?param1=val1¶m2=val2. [109](#), [115–125](#), [127](#), [129](#), [131–134](#), [136–139](#), [141–149](#), [151–155](#)

polling

Sposób komunikacji, w którym **klient** cyklicznie „odpytuje” **serwer** (np. co kilka sekund) o nowe dane, zwykle poprzez [HTTP..](#) [182](#), [212](#)

prefiks

Ustalony początek ścieżki (np. /app lub /subscribe), który porządkuje i rozróżnia typy komunikacji w aplikacji.. [183](#), [184](#), [187](#)

Props

Właściwości przekazywane do komponentu React przez komponent nadzędny; służą do konfiguracji i przekazywania danych. Powinny być traktowane jako tylko do odczytu (read-only) wewnątrz komponentu potomnego.. [163](#)

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). [162](#)

publish–subscribe

Model komunikacji, w którym nadawca publikuje zdarzenia na kanał, a odbiorcy, którzy go **subskrybują**, automatycznie otrzymują wiadomości.. [181](#), [217](#)

ramka

Podstawowa jednostka danych przesyłana w ramach połączenia [WebSocket..](#) [181](#), [182](#)

React

Biblioteka JavaScript do budowy interfejsów użytkownika w oparciu o komponenty deklaratywne i wirtualny DOM. Zapewnia jednokierunkowy przepływ danych oraz zarządzanie stanem komponentów.. [19](#), [100](#), [101](#), [163](#), [195](#)

Reconnect

Automatyczne ponowne nawiązanie połączenia po jego zerwaniu (np. po chwilowej utracie internetu).. [192](#)

Redis

Baza danych typu klucz-wartość wykorzystywana jako szybka warstwa [cache](#). [23](#), [100](#), [101](#), [103](#)

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.. [163](#), [164](#), [187](#), [197–199](#)

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.. [169](#), [172](#)

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 serwrem, 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.

[25](#), [101](#), [106](#), [109](#), [181](#), [182](#), [211](#)

Review kodu

Proces polegający na wzajemnym przeglądzie kodu źródłowego przez programistów w celu wykrycia błędów, poprawy jakości oraz zwiększenia spójności projektu. [25](#), [202](#)

RFC

Dokument opisujący standardy i zalecenia dotyczące Internetu (ang. *Request for Comments*).. [180](#), [211](#)

Serializacja

Proces zamiany obiektu (np. w TypeScript/Java) na format przenośny, np. JSON, w celu przesłania lub zapisu. [193](#)

serwer

Część systemu (zwykle po stronie [Backendu](#)), która przetwarza żądania i udostępnia dane lub funkcje [klientom..](#) [180–182](#), [184–186](#), [210–213](#), [217](#), [218](#)

Sidebar

Boczny panel w interfejsie użytkownika, zawierający menu nawigacyjne lub dodatkowe opcje funkcjonalne aplikacji. [99](#), [202–204](#)

Singleton

Wzorzec projektowy gwarantujący istnienie jednej, współdzielonej instancji obiektu w aplikacji. [197](#)

SockJS

Mechanizm kompatybilności dla [WebSocket](#): gdy przeglądarka lub sieć nie obsługuje WebSocket, SockJS może użyć alternatywnego sposobu komunikacji, aby zachować podobne zachowanie.. [184](#), [191](#), [192](#)

SPA

(ang. *Single Page Application*); aplikacja webowa, która zawiera jeden plik HTML. Po jednorazowym wczytaniu jej zawartość jest zmieniana bez ponownego przeładowania. [19](#)

Spot

Spotkanie zespołu projektowego, zazwyczaj krótkie i regularne, służące omówieniu postępów prac, problemów oraz planów na najbliższy okres. [25](#), [175–180](#)

Spring Framework

Framework, który służy do tworzenia aplikacji w Javie. Zajmuje się między innymi ustawianiem konfiguracji oraz zarządzaniem zależnościami, np.: wstrzykiwaniem

(ang. *Dependency Injection*), odwróceniem kontroli (ang. *Inversion of Control*). Oferuje wiele modułów wspierających, które ułatwiają rozwój projektów. [16](#), [180](#), [182](#)

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).. [163](#), [187](#), [197](#), [198](#)

STOMP

Prosty protokół wiadomościowy działający „nad” [WebSocket](#). Wprowadza pojęcia [destynacji](#), [subskrypcji](#) i wysyłania komunikatów w modelu [publish–subscribe..](#) [180–183](#), [185](#), [186](#), [191–193](#), [195](#), [196](#), [198](#), [207](#), [209](#)

StompJS

Biblioteka JavaScript/TypeScript (np. `@stomp/stompjs`) implementująca klienta protokołu STOMP. [192](#)

Store (Redux)

Centralne miejsce przechowywania stanu aplikacji w Redux. Pozwala odczytywać aktualny stan oraz wysyłać akcje zmieniające ten stan.. [199](#)

subskrypcja

Mechanizm „zapisania się” na kanał ([destynacje](#)) w celu automatycznego odbierania wiadomości publikowanych przez serwer.. [181](#), [183](#), [185–187](#), [192](#), [195](#), [198](#), [209](#), [213](#), [217](#)

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”.. [25](#)

tablica zależności

Lista wartości (dependencies) sterująca ponownym uruchamianiem `useEffect` w React; zmiana zależności powoduje wykonanie sprzątania i ponowne uruchomienie efektu. [195](#)

TypeScript

Rozszerzenie do języka JavaScript dodający statyczne typowanie, interfejsy i narzędzia do większych projektów. Kompiluje się do czystego JavaScript, ułatwiając wykrywanie błędów w czasie komplikacji i refaktoryzacji.. [100](#), [101](#), [163](#), [188–191](#), [194](#), [195](#)

UI

Interfejs użytkownika (ang. *User Interface*); warstwa prezentacji odpowiedzialna za sposób wyświetlania danych oraz interakcji użytkownika z aplikacją.. [19](#), [26](#), [163](#)

UML

(ang. *Unified Modeling Language*); graficzny język wizualizacji, specyfikowania oraz dokumentowania składników systemów informatycznych. . [26](#)

useEffect

Hook React służący do uruchamiania efektów ubocznych po renderowaniu (oraz do definiowania funkcji sprzątającej wykonywanej przy odmontowaniu lub zmianie zależności). [195](#)

WebSocket

Protokół komunikacyjny umożliwiający utrzymanie stałego połączenia pomiędzy klientem a serwerem oraz dwukierunkową wymianę danych w czasie zblizonym do rzeczywistego.. [5](#), [180–184](#), [187](#), [191](#), [195](#), [196](#), [210](#), [213](#), [216](#), [217](#)

Wolumen

Zarządzane przez Dockera magazyny danych dla kontenerów, które są odizolowane od maszyny hosta.. [24](#)

zmienna środowiskowa

Parametr konfiguracyjny przekazywany aplikacji z zewnątrz (np. plik `.env` lub konfiguracja CI/CD), pozwalający zmieniać zachowanie bez modyfikacji kodu. [196](#)

Spis tabel

| | |
|--------------------------------------------------------------------------------------|----|
| Tabela 2.1: Karta udziałowca: Zespół projektowy | 8 |
| Tabela 2.2: Karta udziałowca: Promotor | 9 |
| Tabela 2.3: Karta udziałowca: Droniarze | 10 |
| | |
| Tabela 3.1: Zestawienie wszystkich bibliotek użytych na backendzie. | 19 |
| Tabela 3.2: Zestawienie wszystkich bibliotek i pluginów użytych na fron- | |
| tendzie. | 23 |
| Tabela 3.3: Usługa zewnętrzna: GitHub Actions (CI) | 28 |
| Tabela 3.4: Usługa zewnętrzna: Azure Blob Storage | 28 |
| Tabela 3.5: Usługa zewnętrzna: Mailtrap | 28 |
| Tabela 3.6: Usługa zewnętrzna: LocationIQ | 28 |
| Tabela 3.7: Usługa zewnętrzna: Google Maps (Maps URLs) | 29 |
| Tabela 3.8: Usługa zewnętrzna: OpenFreeMap | 29 |
| Tabela 3.9: Usługa zewnętrzna: Open-Meteo | 29 |
| Tabela 3.10: Usługa zewnętrzna: Tenor GIF API | 30 |
| Tabela 3.11: Usługa zewnętrzna: Where the ISS at? | 30 |
| | |
| Tabela 4.1: Scenariusz przypadku użycia: Rejestracja użytkownika | 43 |
| Tabela 4.2: Scenariusz przypadku użycia: Logowanie użytkownika | 44 |
| Tabela 4.3: Scenariusz przypadku użycia: Wykupienie subskrypcji premium | 45 |
| Tabela 4.4: Scenariusz przypadku użycia: Resetowanie hasła | 46 |
| Tabela 4.5: Scenariusz przypadku użycia: Zmiana hasła w ustawieniach konta | 47 |
| Tabela 4.6: Scenariusz przypadku użycia: Wylogowanie użytkownika | 48 |
| Tabela 4.7: Scenariusz przypadku użycia: Przeglądanie powiadomień . . . | 48 |

| | |
|-------------------------------------------------------------------------------------------------|----|
| Tabela 4.8: Scenariusz przypadku użycia: Przeszukiwanie historii czatu | 49 |
| Tabela 4.9: Scenariusz przypadku użycia: Przeglądanie wysłanych plików na czacie | 50 |
| Tabela 4.10: Scenariusz przypadku użycia: Zmiana typu mapy | 51 |
| Tabela 4.11: Scenariusz przypadku użycia: Przeglądanie stref PANSA | 52 |
| Tabela 4.12: Scenariusz przypadku użycia: Wyszukiwanie spota w globalnej wyszukiwarce | 53 |
| Tabela 4.13: Scenariusz przypadku użycia: Przejście do spota na mapie z wyszukiwarki | 54 |
| Tabela 4.14: Scenariusz przypadku użycia: Przeglądanie mapy spotów | 54 |
| Tabela 4.15: Scenariusz przypadku użycia: Otwarcie szczegółów spota | 55 |
| Tabela 4.16: Scenariusz przypadku użycia: Przeglądanie komentarzy do spota | 56 |
| Tabela 4.17: Scenariusz przypadku użycia: Przeglądanie pogody na spocie | 57 |
| Tabela 4.18: Scenariusz przypadku użycia: Wyszukiwanie spota na mapie | 58 |
| Tabela 4.19: Scenariusz przypadku użycia: Utworzenie prywatnego czatu | 58 |
| Tabela 4.20: Scenariusz przypadku użycia: Otworzenie czatu | 59 |
| Tabela 4.21: Scenariusz przypadku użycia: Utworzenie czatu grupowego | 60 |
| Tabela 4.22: Scenariusz przypadku użycia: Przeglądanie listy czatów | 61 |
| Tabela 4.23: Scenariusz przypadku użycia: Wysyłanie wiadomości na czacie | 62 |
| Tabela 4.24: Scenariusz przypadku użycia: Wysyłanie GIF-a na czacie | 63 |
| Tabela 4.25: Scenariusz przypadku użycia: Wysyłanie pliku na czacie | 64 |
| Tabela 4.26: Scenariusz przypadku użycia: Edycja ustawień czatu | 65 |
| Tabela 4.27: Scenariusz przypadku użycia: Dodanie członka do czatu grupowego | 65 |
| Tabela 4.28: Scenariusz przypadku użycia: Przeglądanie postów na forum | 66 |
| Tabela 4.29: Scenariusz przypadku użycia: Wyszukiwanie postów na forum | 67 |
| Tabela 4.30: Scenariusz przypadku użycia: Dodanie posta na forum | 69 |
| Tabela 4.31: Scenariusz przypadku użycia: Dodanie komentarza na forum | 69 |
| Tabela 4.32: Scenariusz przypadku użycia: Przeglądanie historii interakcji z postami | 70 |

| | |
|---------------------------------------------------------------------------------------------------|----|
| Tabela 4.33: Scenariusz przypadku użycia: Zarządzanie komentarzami na forum | 71 |
| Tabela 4.34: Scenariusz przypadku użycia: Zgłoszenie komentarza naruszającego regulamin | 72 |
| Tabela 4.35: Scenariusz przypadku użycia: Zgłoszenie posta na forum | 73 |
| Tabela 4.36: Scenariusz przypadku użycia: Przeglądanie komentarzy pod postem | 74 |
| Tabela 4.37: Scenariusz przypadku użycia: Dodanie spota w panelu użytkownika | 75 |
| Tabela 4.38: Scenariusz przypadku użycia: Przeglądanie profilu użytkownika | 76 |
| Tabela 4.39: Scenariusz przypadku użycia: Przeglądanie profilu innego użytkownika | 77 |
| Tabela 4.40: Scenariusz przypadku użycia: Dodanie użytkownika do znajomych | 78 |
| Tabela 4.41: Scenariusz przypadku użycia: Przeglądanie społeczności | 78 |
| Tabela 4.42: Scenariusz przypadku użycia: Przeglądanie dodanych spotów | 79 |
| Tabela 4.43: Scenariusz przypadku użycia: Edycja danych użytkownika | 80 |
| Tabela 4.44: Scenariusz przypadku użycia: Przeglądanie dodanych zdjęć do spotów | 81 |
| Tabela 4.45: Scenariusz przypadku użycia: Przeglądanie dodanych filmów do spotów | 82 |
| Tabela 4.46: Scenariusz przypadku użycia: Przeglądanie dodanych komentarzy do spotów | 83 |
| 4.47 Profil użytkownika | 84 |
| 4.48 Lista dodanych spotów | 85 |
| 4.49 Dodanie spota | 86 |
| 4.50 Lista zdjęć | 87 |
| 4.51 Lista filmów | 87 |
| 4.52 Lista znajomych | 88 |
| 4.53 Lista obserwujących | 88 |

| | |
|----------------------------------------------------------------------------------------|-----|
| 4.54 Lista obserwowanych | 89 |
| 4.55 Lista polubionych/odwiedzonych/planowanych spotów | 89 |
| 4.56 Lista komentarzy | 90 |
| 4.57 Ustawienia profilu | 91 |
| 4.58 Resetowanie hasła | 92 |
| 4.59 Dodawanie do znajomych | 93 |
| 4.60 Logowanie i rejestracja | 94 |
| 4.61 Strona główna — podstawowe filtry | 95 |
| 4.62 Strona główna — zaawansowane filtry | 96 |
| 4.63 Ustawienia motywu (ręczna zmiana) | 97 |
| 4.64 Zapamiętanie preferencji motywu | 98 |
| 4.65 Szybki przełącznik motywu w interfejsie | 99 |
| Tabela 7.1: Zestawienie endpointów: panelu użytkownika | 110 |
| Tabela 7.2: Zestawienie endpointów: modułu spotów | 111 |
| Tabela 7.3: Zestawienie endpointów: komentarzy do spotów | 112 |
| Tabela 7.4: Zestawienie endpointów: postów forum | 112 |
| Tabela 7.5: Zestawienie endpointów: komentarzy forum | 113 |
| Tabela 7.6: Zestawienie endpointów: konta użytkownika i autoryzacji | 113 |
| Tabela 7.7: Zestawienie endpointów: integracji GIF-ów | 114 |
| Tabela 7.8: Zestawienie endpointów: modułu czatu | 114 |
| Tabela 7.9: Karta endpointu: /public/user-dashboard/profile/{targetUsername} | 115 |
| Tabela 7.10: Karta endpointu: /user-dashboard/favorite-spots | 116 |
| Tabela 7.11: Karta endpointu: /user-dashboard/photos | 117 |
| Tabela 7.12: Karta endpointu: /user-dashboard/add-spot | 118 |
| Tabela 7.13: Karta endpointu: /user-dashboard/add-spot | 119 |
| Tabela 7.14: Karta endpointu: /public/spot/gallery | 120 |
| Tabela 7.15: Karta endpointu: /public/spot/current-view | 121 |
| Tabela 7.16: Karta endpointu: /public/spot/get-spot-basic-weather | 122 |
| Tabela 7.17: Karta endpointu: /public/spot/get-spot-detailed-weather | 123 |
| Tabela 7.18: Karta endpointu: /public/spot/get-spot-wind-speeds | 124 |
| Tabela 7.19: Karta endpointu: /public/spot/most-popular | 125 |

| | |
|--------------------------------------------------------------------------|-----|
| Tabela 7.20: Karta endpointu: /public/spot/search/home-page | 126 |
| Tabela 7.21: Karta endpointu: /public/spot/search/home-page/locations | 127 |
| Tabela 7.22: Karta endpointu: /public/spot/search/home-page/advance . | 128 |
| Tabela 7.23: Karta endpointu: /public/spot/{spotId}/comments | 130 |
| Tabela 7.24: Karta endpointu: /public/spot/{spotId}/comments/{commentId} | 131 |
| Tabela 7.25: Karta endpointu: /spot/{spotId}/comments | 132 |
| Tabela 7.26: Karta endpointu: /spot/comments/{commentId}/vote . . . | 133 |
| Tabela 7.27: Karta endpointu: /spot/comments/vote-type | 133 |
| Tabela 7.28: Karta endpointu: /public/post/{postId} | 135 |
| Tabela 7.29: Karta endpointu: /post | 136 |
| Tabela 7.30: Karta endpointu: /post/{postId} | 137 |
| Tabela 7.31: Karta endpointu: /post/{postId}/vote | 137 |
| Tabela 7.32: Karta endpointu: /public/categories-tags | 138 |
| Tabela 7.33: Karta endpointu: /public/post/{postId}/comments | 140 |
| Tabela 7.34: Karta endpointu: /post/{postId}/comments | 141 |
| Tabela 7.35: Karta endpointu: /post/comments/{commentId} | 142 |
| Tabela 7.36: Karta endpointu: /post/comments/{commentId}/vote . . . | 142 |
| Tabela 7.37: Karta endpointu: /comments/{commentId}/replies | 143 |
| Tabela 7.38: Karta endpointu: /public/account/register | 144 |
| Tabela 7.39: Karta endpointu: /public/account/login | 144 |
| Tabela 7.40: Karta endpointu: /public/account/forgot-password | 145 |
| Tabela 7.41: Karta endpointu: /public/account/set-new-password . . . | 146 |
| Tabela 7.42: Karta endpointu: /account/check | 146 |
| Tabela 7.43: Karta endpointu: /account/login-success | 147 |
| Tabela 7.44: Karta endpointu: /gifs/trending | 148 |
| Tabela 7.45: Karta endpointu: /gifs/search | 149 |
| Tabela 7.46: Karta endpointu: /chats/{chatId}/messages | 150 |
| Tabela 7.47: Karta endpointu: /chats/get-or-create-private-chat . . . | 151 |
| Tabela 7.48: Karta endpointu: /chats/{chatId}/send-files | 152 |
| Tabela 7.49: Karta endpointu: /chats/create/group | 153 |
| Tabela 7.50: Karta endpointu: /chats/{chatId} | 154 |

Tabela 7.51: Karta endpointu: /chats/group-chat/add/search/{chatId} . 155

Bibliografia

- [1] *Disciplined Agile Delivery*. PMI. 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*. URL: <https://wolski.pro/2024/05/10-wskazovek-poprawiajacych-modelowanie-procesow-biznesowych-w-notacji-bpmn/> (dostęp 19.11.2025).
- [5] *About billing for GitHub Actions*. GitHub Docs. 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. URL: <https://learn.microsoft.com/azure/storage/blobs/scalability-targets> (dostęp 2.11.2025).
- [7] *What are the limitations in Mailtrap?* Mailtrap Docs. URL: <https://help.mailtrap.io/article/111-what-are-the-limitations-in-mailtrap/> (dostęp 2.11.2025).
- [8] *LocationIQ Pricing*. LocationIQ. URL: <https://locationiq.com/pricing> (dostęp 2.11.2025).
- [9] *Google Maps (Maps URLs)*. Google Maps. URL: <https://developers.google.com/maps/documentation/urls/get-started?hl=pl> (dostęp 2.11.2025).
- [10] *OpenFreeMap Documentation*. OpenFreeMap. URL: <https://openfreemap.org/docs> (dostęp 2.11.2025).

- [11] *Open-Meteo API Usage & Pricing*. Open-Meteo. URL: <https://open-meteo.com/en/docs/usage-and-pricing> (dostęp 2.11.2025).
- [12] *Tenor API — Documentation*. Tenor. URL: <https://tenor.com/gifapi/documentation> (dostęp 2.11.2025).
- [13] *Where the ISS at? API*. wheretheiss.at. URL: <https://wheretheiss.at/> (dostęp 2.11.2025).
- [14] *React useState*. URL: <https://react.dev/reference/react/useState> (dostęp 3.11.2025).
- [15] *Redux*. URL: <https://redux.js.org/> (dostęp 3.11.2025).
- [16] *Axios*. URL: <https://axios-http.com/> (dostęp 3.11.2025).
- [17] *Tanstack Query*. URL: <https://tanstack.com/query/latest> (dostęp 3.11.2025).
- [18] *Tailwind*. URL: <https://tailwindcss.com/> (dostęp 3.11.2025).
- [19] *Motion*. URL: <https://motion.dev/> (dostęp 3.11.2025).
- [20] *RFC 6455: The WebSocket Protocol*. IETF. URL: <https://datatracker.ietf.org/doc/html/rfc6455> (dostęp 16.12.2025).
- [21] *WebSocket (Spring Framework Reference Documentation)*. Spring. URL: <https://docs.spring.io/spring-framework/reference/web/websocket.html> (dostęp 16.12.2025).
- [22] *STOMP over WebSocket (Spring Framework Reference Documentation)*. Spring. URL: <https://docs.spring.io/spring-framework/reference/web/websocket/stomp.html> (dostęp 16.12.2025).
- [23] *sockjs-client (WebSocket emulation — JavaScript client)*. SockJS. URL: <https://github.com/sockjs/sockjs-client> (dostęp 16.12.2025).

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.