



POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

Wydział Informatyki

Filia w Gdańsku

Langmesser Adam

Nr albumu s27119

Nazwa specjalizacji: Aplikacje Internetowe

Redosz Mateusz

Nr albumu s27094

Nazwa specjalizacji: Aplikacje Internetowe

Oziemczuk Stanisław

Nr albumu s26982

Nazwa specjalizacji: Aplikacje Internetowe

Badek Kacper

Nr albumu s29168

Nazwa specjalizacji: Aplikacje Internetowe

Aplikacja webowa: spoty-na-drony.pl

Rodzaj pracy

inżynierska

Imię i nazwisko promotora

mgr Adam Urbanowicz

Gdańsk, miesiąc, 2100 obrony

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

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

Słowa kluczowe: — brak —



POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

Karta projektu

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

Data ukończenia projektu: 3 listopada 2025	Recenzent: dr Elżbieta Puźniakowska-Gałuch
--	--

Spis treści

1	Wstęp	6
1.1	O projekcie	6
1.2	Cel i zakres prac	6
1.3	Geneza pomysłu	6
2	Opis problemu	9
2.1	Rich picture	9
2.2	Udziałowcy	9
2.3	Istniejące rozwiązania	9
2.4	Wizja rozwiązania	9
2.5	Aspekty społeczne i biznesowe	9
2.5.1	Aspekty społeczne	9
2.5.2	Aspekty biznesowe	9
3	Planowanie	10
3.1	Metodologia pracy	10
3.2	Harmonogram projektu	10
3.3	Technologie i narzędzia	10
3.3.1	Technologie	10
3.3.2	Narzędzia	10
3.4	Zasoby i ograniczenia	10
3.4.1	Zasoby	10
3.4.2	Ograniczenia	10
3.5	Analiza ryzyka	10

4	Analiza wymagań	11
4.1	Przypadki użycia	12
4.1.1	Aktorzy	12
4.1.2	Diagram przypadków użycia	12
4.1.3	Scenariusz przypadków użycia	12
4.2	Wymagania ogólne i dziedzinowe	12
4.3	Wymagania funkcjonalne	12
4.3.1	Funkcjonalności dla mapy	12
4.3.2	Funkcjonalności dla chatu	12
4.3.3	Funkcjonalności dla forum	12
4.3.4	Funkcjonalności dla konta użytkownika	12
4.3.5	Funkcjonalności dla logowania i rejestracji	19
4.3.6	Funkcjonalności dla wyszukiwarki spotów	20
4.3.7	Funkcjonalności dla motywu	22
4.4	Wymagania pozafunkcjonalne	22
4.5	Wymagania interfejs z otoczeniem	22
4.6	Wymagania na środowisko docelowe	22
5	Projekt	23
5.1	Wzorce projektowe	23
5.2	Architektura systemu	23
5.2.1	Diagram architektury	23
5.2.2	Komponenty systemu	23
5.3	Projekt bazy danych	23
5.3.1	Model danych	23
5.3.2	Diagram ERD	23
5.4	Architektura interfejsu użytkownika	23
5.4.1	Projekt strony głównej	23
5.4.2	Projekt panelu logowania	23
5.4.3	Projekt mapy	23
5.4.4	Projekt chatu	23
5.4.5	Projekt forum	23

5.4.6	Projekt konta użytkownika	23
6	Przebieg realizacji projektu	24
6.1	Sprint 1	24
6.2	Sprint 2	24
7	Realizacja Projektu	25
7.1	Implementacja backendu	25
7.1.1	Struktura projektu	25
7.1.2	Endpointy systemu	25
7.1.3	Integracja z bazą danych	28
7.1.4	Obsługa uwierzytelnienia	28
7.1.5	Konteneryzacja	28
7.2	Implementacja frontendu	28
7.2.1	Struktura aplikacji	28
7.2.2	Zarządzanie stanem i przepływ danych	33
7.2.3	Integracja i komunikacja z backendem	36
7.2.4	Style	36
7.2.5	Strona główna	36
7.2.6	Mapa	36
7.2.7	Chat	36
7.2.8	Forum	36
7.2.9	Konto użytkownika	36
7.2.10	Panel logowania	36
7.3	Implementacja CI/CD	36
8	Testy	37
8.1	Testy jednostkowe	37
8.2	Testy integracyjne	37
8.3	Testy E2E	37
8.4	Wyniki testów i wnioski	37

9	Prezentacja systemu	38
9.1	Strona główna	38
9.2	Strona mapy	38
9.3	Strona chatu	38
9.4	Strona forum	38
9.5	Panel logowania	38
9.6	Panel konta użytkownika	38
10	Nakład pracy	39
10.1	Ogólny nakład pracy	39
10.2	Indywidualne nakłady pracy	39
10.2.1	Adam Langmesser	39
10.2.2	Mateusz Redosz	39
10.2.3	Stanisław Oziemczuk	42
10.2.4	Kacper Badek	42
11	Podsumowanie	43
11.1	Osiągnięte rezultaty	43
11.2	Napotkane wyzwania	43
11.3	Plany na przyszłość	43

Rozdział 1

Wstęp

1.1 O projekcie

1.2 Cel i zakres prac

1.3 Geneza pomysłu

Słownik pojęć i skrótów

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, 40

Bibliote Zewnętrzny lub wewnętrzny zestaw gotowych funkcji, klas, komponentów lub modułów, który można wielokrotnie wykorzystywać w projekcie zamiast pisać wszystko od zera.. 33

CI/CD Skrót od *Continuous Integration/Continuous Deployment*. Praktyka programistyczna polegająca na automatyzacji procesu budowania, testowania i wdrażania oprogramowania.. 39, 42

Design Etap lub proces projektowania wyglądu i funkcjonalności aplikacji, obejmujący zarówno aspekty wizualne, jak i użytkowe (UX/UI).. 39

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.. 28

Framework Zestaw narzędzi, bibliotek i struktur wspomagających tworzenie aplikacji. Ułatwia programowanie poprzez dostarczenie gotowych komponentów oraz określenie zasad organizacji kodu.. 2

Frontend Warstwa aplikacji odpowiedzialna za interfejs użytkownika oraz interakcję z użytkownikiem. Zazwyczaj tworzona przy użyciu technologii takich jak HTML, CSS i JavaScript.. 2, 40

JWT Skrót od *JSON Web Token*. Standard służący do bezpiecznego przekazywania informacji między stronami w formacie JSON, często używany w procesach autoryzacji użytkowników.. 39

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.. 32

Review kodu Proces polegający na wzajemnym przeglądzie kodu źródłowego przez programistów w celu wykrycia błędów, poprawy jakości oraz zwiększenia spójności projektu.. 39

Sidebar Boczny panel w interfejsie użytkownika, zawierający menu nawigacyjne lub dodatkowe opcje funkcjonalne aplikacji.. 39–41

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).. 33

UI Interfejs użytkownika (ang. *User Interface*); warstwa prezentacji odpowiedzialna za sposób wyświetlania danych oraz interakcji użytkownika z aplikacją.. 33

Rozdział 2

Opis problemu

2.1 Rich picture

2.2 Udziałowcy

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
- 3.2 Harmonogram projektu
- 3.3 Technologie i narzędzia
 - 3.3.1 Technologie
 - 3.3.2 Narzędzia
- 3.4 Zasoby i ograniczenia
 - 3.4.1 Zasoby
 - 3.4.2 Ograniczenia
- 3.5 Analiza ryzyka

Rozdział 4

Analiza wymagań

4.1 Przypadki użycia

4.1.1 Aktorzy

4.1.2 Diagram przypadków użycia

4.1.3 Scenariusz przypadków użycia

4.2 Wymagania ogólne i dziedzinowe

4.3 Wymagania funkcjonalne

4.3.1 Funkcjonalności dla mapy

4.3.2 Funkcjonalności dla chatu

4.3.3 Funkcjonalności dla forum

4.3.4 Funkcjonalności dla konta użytkownika

KARTA WYMAGANIA			
Identyfikator:	jednoznaczny symbol np. FO1, FO2 ..	Priorytet:	M
Nazwa:	Profil użytkownika		
Opis:	Jako użytkownik chcę mieć dostęp do strony profilu, aby sprawdzić informacje o swoim profilu.		
Kryteria akceptacji:	Użytkownik może zobaczyć informacje o ilości znajomych, obserwowanych i obserwujących oraz o najpopularniejszych zdjęciach.		
Dane wejściowe:	Lista zdjęć i liczby znajomych, obserwujących i obserwowanych.		
Warunki początkowe:	Użytkownik jest zalogowany		
Warunki końcowe:	Wyświetlenie się informacji		
Sytuacje wyjątkowe:			
Szczegóły implementacji:			
Udziałowiec:			
Wymagania powiązane:			

Tabela 4.1:

KARTA WYMAGANIA			
Identyfikator:	jednoznaczny symbol np. FO1, FO2 ..	Priorytet:	M
Nazwa:	Lista dodanych Spotów		
Opis:	Jako użytkownik chcę mieć dostęp do sprawdzenia listy dodanych spotów.		
Kryteria akceptacji:	Użytkownik może zobaczyć listę danych spotów.		
Dane wejściowe:	Lista dodanych spotów.		
Warunki początkowe:	Użytkownik jest zalogowany.		
Warunki końcowe:	Wyświetlenie się informacji.		
Sytuacje wyjątkowe:			
Szczegóły implementacji:			
Udziałowiec:			
Wymagania powiązane:			

Tabela 4.2:

KARTA WYMAGANIA			
Identyfikator:	jednoznaczny symbol np. FO1, FO2 ..	Priorytet:	M
Nazwa:	Dodanie Spotów		
Opis:	Jako użytkownik chcę mieć dostęp do formularza do dodania spota.		
Kryteria akceptacji:	Użytkownik ma dostęp do formularza do dodania spota.		
Dane wejściowe:	Formularz		
Warunki początkowe:	Użytkownik jest zalogowany.		
Warunki końcowe:	Wyświetlenie się formularza.		
Sytuacje wyjątkowe:			
Szczegóły implementacji:			
Udziałowiec:			
Wymagania powiązane:			

Tabela 4.3:

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 oraz do komentarzy pod spotem.		
Kryteria akceptacji:	Użytkownik ma dostęp do listy zdjęć.		
Dane wejściowe:	Lista zdjęć.		
Warunki początkowe:	Użytkownik jest zalogowany.		
Warunki końcowe:	Wyświetlenie się informacji.		
Sytuacje wyjątkowe:			
Szczegóły implementacji:			
Udziałowiec:			
Wymagania powiązane:			

Tabela 4.4:

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 oraz do komentarzy pod spotem.		
Kryteria akceptacji:	Użytkownik ma dostęp do listy filmów.		
Dane wejściowe:	Lista filmów.		
Warunki początkowe:	Użytkownik jest zalogowany.		
Warunki końcowe:	Wyświetlenie się informacji.		
Sytuacje wyjątkowe:			
Szczegóły implementacji:			
Udziałowiec:			
Wymagania powiązane:			

Tabela 4.5:

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:	Wyświetlenie się informacji.		
Sytuacje wyjątkowe:			
Szczegóły implementacji:			
Udziałowiec:			
Wymagania powiązane:			

Tabela 4.6:

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:	Wyświetlenie się informacji.		
Sytuacje wyjątkowe:			
Szczegóły implementacji:			
Udziałowiec:			
Wymagania powiązane:			

Tabela 4.7:

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:	Wyświetlenie się informacji.		
Sytuacje wyjątkowe:			
Szczegóły implementacji:			
Udziałowiec:			
Wymagania powiązane:			

Tabela 4.8:

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.		
Dane wejściowe:	Lista spotów		
Warunki początkowe:	Użytkownik jest zalogowany.		
Warunki końcowe:	Wyświetlenie się informacji.		
Sytuacje wyjątkowe:			
Szczegóły implementacji:			
Udziałowiec:			
Wymagania powiązane:			

Tabela 4.9:

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 komentarzy.		
Dane wejściowe:	Lista komentarzy		
Warunki początkowe:	Użytkownik jest zalogowany.		
Warunki końcowe:	Wyświetlenie się informacji.		
Sytuacje wyjątkowe:			
Szczegóły implementacji:			
Udziałowiec:			
Wymagania powiązane:			

Tabela 4.10:

KARTA WYMAGANIA			
Identyfikator:	jednoznaczny symbol np. FO1, FO2 ..	Priorytet:	M
Nazwa:	Ustawienia		
Opis:	Jako użytkownik chcę mieć możliwość zmiany danych.		
Kryteria akceptacji:	Użytkownik ma możliwość zmiany danych.		
Dane wejściowe:	Formularz do zmiany danych.		
Warunki początkowe:	Użytkownik jest zalogowany.		
Warunki końcowe:	Wyświetlenie się formularza do edycji.		
Sytuacje wyjątkowe:			
Szczegóły implementacji:			
Udziałowiec:			
Wymagania powiązane:			

Tabela 4.11:

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:	Użytkownik po kliknięciu w odpowiedni link ma możliwość zresetowania hasła do swojego konta w aplikacji.		
Dane wejściowe:	Adres email użytkownika do wysłania linku resetującego.		
Warunki początkowe:	Użytkownik podał poprawny email podany podczas rejestracji.		
Warunki końcowe:	Zresetowanie hasła.		
Sytuacje wyjątkowe:			
Szczegóły implementacji:			
Udziałowiec:			
Wymagania powiązane:			

Tabela 4.12:

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 innych użytkowników do swojej listy znajomych.		
Dane wejściowe:	Dane użytkownika, którego chcemy dodać do znajomych.		
Warunki początkowe:	Zalogowany użytkownik.		
Warunki końcowe:	Znajomi są dodani do listy i widoczni w profilu użytkownika.		
Sytuacje wyjątkowe:			
Szczegóły implementacji:			
Udziałowiec:			
Wymagania powiązane:			

Tabela 4.13:

4.3.5 Funkcjonalności dla logowania i rejestracji

KARTA WYMAGANIA			
Identyfikator:	jednoznaczny symbol np. FO1, FO2 ..	Priorytet:	M
Nazwa:	Logowanie i rejestracja		
Opis:	Jako użytkownik chcę mieć możliwość zalogowania się do aplikacji korzystając z formularza lub poprzez użycie konta w Google lub GitHub.		
Kryteria akceptacji:	Użytkownik może zalogować się do aplikacji zarówno poprzez użycie standardowego formularza jak i przez użycie swojego konta w serwisie Google lub GitHub.		
Dane wejściowe:	Dane użytkownika: email, hasło, nazwa użytkownika.		
Warunki początkowe:	Użytkownik nie zalogowany.		
Warunki końcowe:	Działające formularze rejestracji i logowania oraz możliwość zalogowania za pomocą konta Googla i Githuba.		
Sytuacje wyjątkowe:			
Szczegóły implementacji:			
Udziałowiec:			
Wymagania powiązane:			

Tabela 4.14:

4.3.6 Funkcjonalności dla wyszukiwarki spotów

KARTA WYMAGANIA			
Identyfikator:	jednoznaczny symbol np. FO1, FO2 ..	Priorytet:	M
Nazwa:	Strona główna z podstawowymi filtrami		
Opis:	Jako użytkownik chcę mieć dostęp do strony głównej, która wyświetla najpopularniejsze spoty i listę spotów które można filtrować.		
Kryteria akceptacji:	Użytkownik może zobaczyć listę najpopularniejszych miejsc. Lista zawiera zdjęcia, nazwę miejsca, lokalizację, tagi i opinie. Użytkownik może filtrować miejsca według lokalizacji (kraj, region, miasto).		
Dane wejściowe:	Lokalizacja użytkownika (kraj, region, miasto). Ewentualne dane z bazy miejsc turystycznych.		
Warunki początkowe:	Użytkownik nie musi być zalogowany.		
Warunki końcowe:	Użytkownik widzi popularne miejsca z danego miasta (np. Gdańsk). Użytkownik może przejść do szczegółów danego miejsca.		
Sytuacje wyjątkowe:			
Szczegóły implementacji:			
Udziałowiec:			
Wymagania powiązane:			

Tabela 4.15:

KARTA WYMAGANIA			
Identyfikator:	jednoznaczny symbol np. FO1, FO2 ..	Priorytet:	M
Nazwa:	Strona główna z zaawansowanymi filtrami		
Opis:	Jako użytkownik chcę mieć dostęp do strony głównej, która wyświetla najpopularniejsze spoty i listę spotów które można filtrować i sortować.		
Kryteria akceptacji:	Użytkownik może zobaczyć listę najpopularniejszych miejsc. Lista zawiera zdjęcia, nazwę miejsca, lokalizację i krótki opis. Użytkownik może filtrować miejsca według lokalizacji (kraj, region, miasto), po ocenie oraz sortować po ocenie.		
Dane wejściowe:	Lokalizacja użytkownika (kraj, region, miasto), sortowanie i filtry. Ewentualne dane z bazy miejsc turystycznych.		
Warunki początkowe:	Użytkownik nie musi być zalogowany.		
Warunki końcowe:	Użytkownik widzi popularne miejsca z danego miasta (np. Gdańsk). Użytkownik może przejść do szczegółów danego miejsca.		
Sytuacje wyjątkowe:			
Szczegóły implementacji:			
Udziałowiec:			
Wymagania powiązane:			

Tabela 4.16:

4.3.7 Funkcjonalności dla motywu

KARTA WYMAGANIA			
Identyfikator:	jednoznaczny symbol np. FO1, FO2 ..	Priorytet:	M
Nazwa:	Ustawienia motywu.		
Opis:	Jako użytkownik chcę móc zmienić motyw aplikacji.		
Kryteria akceptacji:	Użytkownik może zmieniać ustawienia wyglądu (motywu) aplikacji na jasny lub ciemny.		
Dane wejściowe:	Preferencje użytkownika dotyczące motywu.		
Warunki początkowe:	Zalogowany użytkownik.		
Warunki końcowe:	Zmiana motywu widoczna natychmiast po kliknięciu przycisku.		
Sytuacje wyjątkowe:			
Szczegóły implementacji:			
Udziałowiec:			
Wymagania powiązane:			

Tabela 4.17:

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

5.2.1 Diagram architektury

5.2.2 Komponenty systemu

5.3 Projekt bazy danych

5.3.1 Model danych

5.3.2 Diagram ERD

5.4 Architektura interfejsu użytkownika

5.4.1 Projekt strony głównej

5.4.2 Projekt panelu logowania

5.4.3 Projekt mapy

5.4.4 Projekt chatu

5.4.5 Projekt forum

5.4.6 Projekt konta użytkownika

Rozdział 6

Przebieg realizacji projektu

6.1 Sprint 1

6.2 Sprint 2

Rozdział 7

Realizacja Projektu

7.1 Implementacja backendu

7.1.1 Struktura projektu

7.1.2 Endpointy systemu

GET /user-dashboard/profile

Opis: Zwraca profil aktualnie zalogowanego użytkownika.

Metoda: GET /user-dashboard/profile

Zwraca (200 OK): application/json — obiekt UserProfileDto.

Błąd (404 Not Found): text/plain —

komunikat z wyjątku UserNotFoundByUsernameException.

Przykładowa odpowiedź (200 OK):

```
1  {
2    "username": "john_doe",
3    "profilePhoto": "https://cdn.example.com/profiles/john_doe.
4      jpg",
5    "followersCount": 125,
6    "followedCount": 87,
7    "friendsCount": 32,
8    "photosCount": 58,
9    "mostPopularPhotos": [
10     {
11       "src": "https://cdn.example.com/photos/123.jpg",
12       "heartsCount": 240,
```

```
12     "viewsCount": 3400,
13     "title": "Sunset at the beach",
14     "id": 123
15   }
16 ]
17 }
```

Example response (404 Not Found):

Panel użytkownika

- GET /user-dashboard/profile
- GET /public/user-dashboard/profile/"targetUsername"
- PATCH /user-dashboard/profile
- GET /user-dashboard/friends
- GET /public/user-dashboard/friends/"targetUsername"
- PATCH /user-dashboard/friends
- PATCH /user-dashboard/friends/change-status
- GET /user-dashboard/followers
- GET /public/user-dashboard/followers/"targetUsername"
- GET /user-dashboard/followed
- GET /public/user-dashboard/followed/"targetUsername"
- GET /user-dashboard/friends/find
- GET /user-dashboard/friends/invites
- PATCH /user-dashboard/followed
- GET /user-dashboard/favorite-spots
- PATCH /user-dashboard/favorite-spots

- 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

Strona główna

- 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

Konto użytkownika

- 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

7.1.3 Integracja z bazą danych

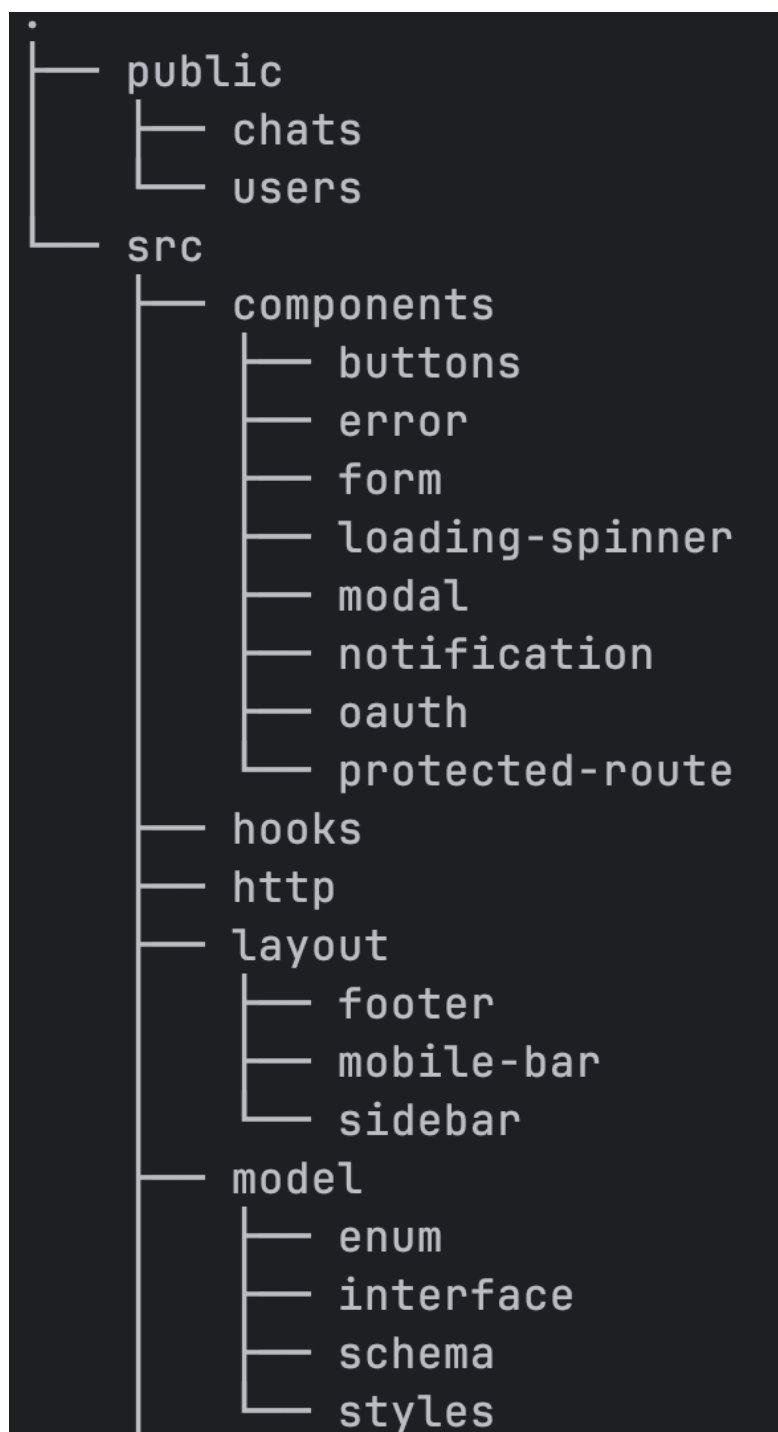
7.1.4 Obsługa uwierzytelnienia

7.1.5 Konteneryzacja

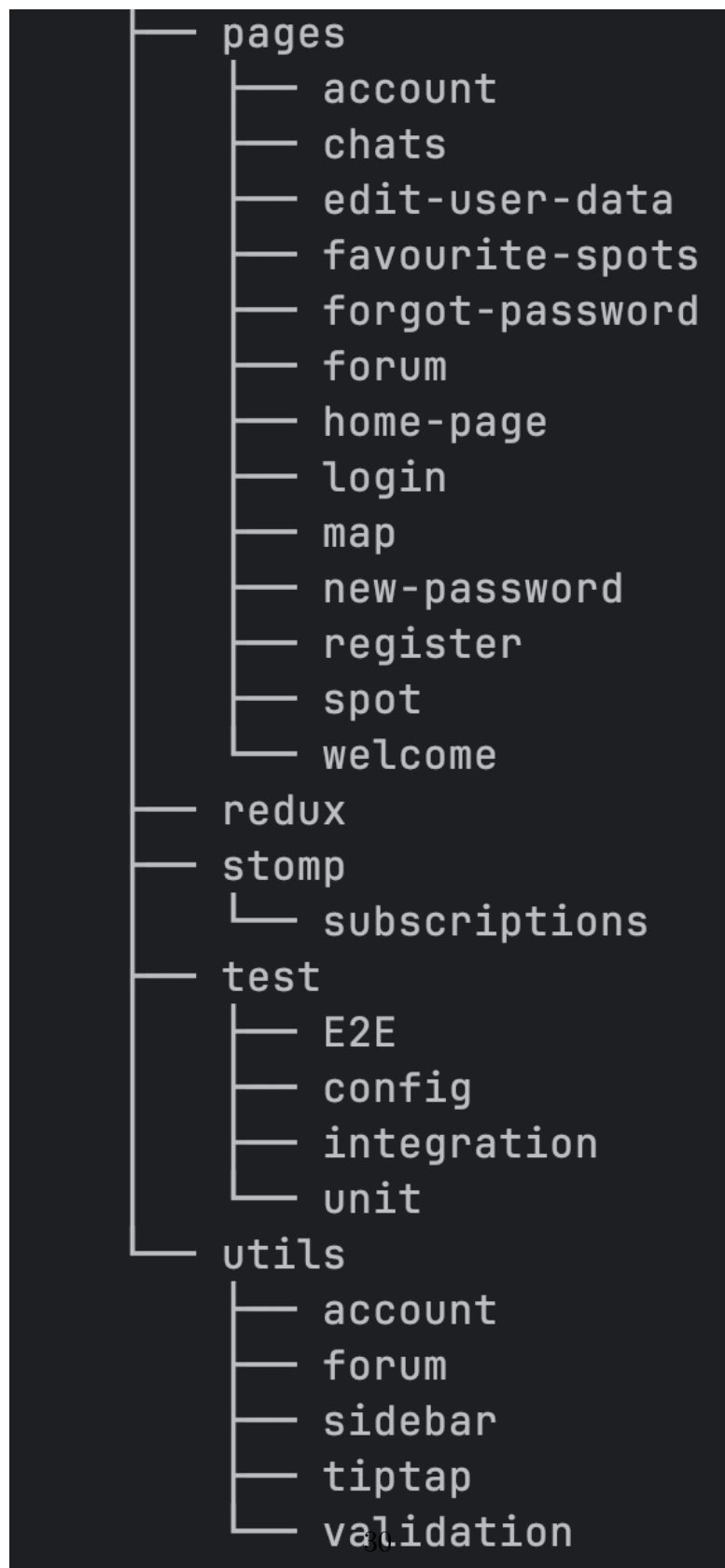
7.2 Implementacja frontendu

7.2.1 Struktura aplikacji

Architektura aplikacji frontendowej została zaprojektowana w strukturze Folder by type, która polega na podziale kodu według typu zasobu (komponenty, strony, modele itd.). Każdy plik znajduje się w katalogu odpowiadającym jego przeznaczeniu, co widać na rysunkach 7.1 oraz 7.2.



Rysunek 7.1: Struktura katalogów (1)



Rysunek 7.2: Struktura katalogów (2)

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

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

Rysunek 7.3: Implementacja routera (1)

```

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

export default router;

```

Rysunek 7.4: Implementacja routera (2)

W projekcie zastosowano również wzorzec Protected route, który służy do zabezpieczania wybranych tras przed dostępem użytkowników niezalogowanych. W pliku `router.tsx`, znajdującym się w głównym katalogu projektu, w konfiguracji przekazywanej do funkcji `createBrowserRouter` (rysunki 7.3 oraz 7.4), wybrane

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

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

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

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

Rysunek 7.5: Implementacja komponentu bramki (`ProtectedRoute`)

7.2.2 Zarządzanie stanem i przepływ danych

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

Do przechowywania Stanu lokalnego, ograniczonego tylko do danego komponentu (lub jego najbliższych elementów podrzędnych), wykorzystujemy hooki `useState` oraz `useEffect`. Natomiast w przypadku bardziej złożonej logiki lub potrzeby ponownego wykorzystania kodu powstały hooki niestandardowe, takie jak `useScreenSize`, `useDarkMode` czy `useClickOutside`. Dzięki temu większość logiki biznesowej została wydzielona z warstwy UI, co poprawia czytelność i ułatwia utrzymanie kodu.

Z racji tego, że korzystamy z Reacta w połączeniu z TypeScriptem, przygotowaliśmy również własne hooki wspomagające typowanie, takie jak `useDispatchTyped` oraz `useSelectorTyped`. Pozwalają one na bezpieczne typowanie akcji oraz selek-

torów Reduxa bez konieczności powtarzania adnotacji typów w każdym komponencie. Fragmenty tej implementacji przedstawiono na rysunkach 7.6 oraz 7.7.

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

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

Rysunek 7.6: Konfiguracja sklepu (Redux store)

```

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

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

export const accountSlice : Slice<AccountSliceProps, { setIsLogged(st... = createSlice({ Show usages  Mredosz +1
  name: "account",
  initialState,
  reducers: {
    setIsLogged(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<{ setIsLogged(state: W... = accountSlice.actions; Show usages  Mredosz

```

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

7.2.3 Integracja i komunikacja z backendem

7.2.4 Style

7.2.5 Strona główna

7.2.6 Mapa

7.2.7 Chat

7.2.8 Forum

7.2.9 Konto użytkownika

7.2.10 Panel logowania

7.3 Implementacja CI/CD

Rozdział 8

Testy

8.1 Testy jednostkowe

8.2 Testy integracyjne

8.3 Testy E2E

8.4 Wyniki testów i wnioski

Rozdział 9

Prezentacja systemu

9.1 Strona główna

9.2 Strona mapy

9.3 Strona chatu

9.4 Strona forum

9.5 Panel logowania

9.6 Panel konta użytkownika

Rozdział 10

Nakład pracy

10.1 Ogólny nakład pracy

10.2 Indywidualne nakłady pracy

10.2.1 Adam Langmesser

10.2.2 Mateusz Redosz

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

1. Dokumentacja

- TODO

2. Design

- Ustalić paletę kolorystyczną

- Propozycja wyglądu

3. Backend i Frontend

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

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

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

4. CI/CD

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

5. Praca dyplomowa

- Uzupełnienie informacji o zespole i podział na rozdziały

10.2.3 Stanisław Oziemczuk

10.2.4 Kacper Badek

Rozdział 11

Podsumowanie

- 11.1 Osiągnięte rezultaty
- 11.2 Napotkane wyzwania
- 11.3 Plany na przyszłość

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.