



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: 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 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: spotty-na-drony.pl Temat projektu po angielsku: Web application: spotty-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: 10 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 | 7 |
| 2.1 Rich picture | 7 |
| 2.2 Udziałowcy | 7 |
| 2.3 Istniejące rozwiązania | 7 |
| 2.4 Wizja rozwiązania | 7 |
| 2.5 Aspekty społeczne i biznesowe | 7 |
| 2.5.1 Aspekty społeczne | 7 |
| 2.5.2 Aspekty biznesowe | 7 |
| 3 Planowanie | 8 |
| 3.1 Metodologia pracy | 8 |
| 3.1.1 Przegląd rozważanych podejść | 8 |
| 3.1.2 Odrzucone podejścia | 8 |
| 3.1.3 Wybrane podejście: Disciplined Agile Delivery (Lean Life Cycle) | 9 |
| 3.1.4 Narzędzia i komunikacja | 9 |
| 3.1.5 Podział ról w zespole | 10 |
| 3.2 Harmonogram projektu | 10 |
| 3.3 Technologie i narzędzia | 12 |
| 3.3.1 Technologie | 12 |

| | | |
|----------|---|-----------|
| 3.3.2 | Narzędzia | 12 |
| 3.4 | Zasoby i ograniczenia | 15 |
| 3.4.1 | Zasoby | 15 |
| 3.4.2 | Ograniczenia | 15 |
| 3.5 | Analiza ryzyka | 15 |
| 4 | Analiza wymagań | 16 |
| 4.1 | Przypadki użycia | 17 |
| 4.1.1 | Aktorzy | 17 |
| 4.1.2 | Diagram przypadków użycia | 17 |
| 4.1.3 | Scenariusz przypadków użycia | 17 |
| 4.2 | Wymagania ogólne i dziedzinowe | 17 |
| 4.3 | Wymagania funkcjonalne | 17 |
| 4.3.1 | Funkcjonalności dla mapy | 17 |
| 4.3.2 | Funkcjonalności dla chatu | 17 |
| 4.3.3 | Funkcjonalności dla forum | 17 |
| 4.3.4 | Funkcjonalności dla konta użytkownika | 17 |
| 4.3.5 | Funkcjonalności dla logowania i rejestracji | 17 |
| 4.3.6 | Funkcjonalności dla wyszukiwarki spotów | 17 |
| 4.3.7 | Funkcjonalności dla motywu | 17 |
| 4.4 | Wymagania pozafunkcjonalne | 17 |
| 4.5 | Wymagania interfejs z otoczeniem | 17 |
| 4.6 | Wymagania na środowisko docelowe | 17 |
| 5 | Projekt | 18 |
| 5.1 | Wzorce projektowe | 18 |
| 5.2 | Architektura systemu | 18 |
| 5.2.1 | Diagram architektury | 18 |
| 5.2.2 | Komponenty systemu | 18 |
| 5.3 | Projekt bazy danych | 18 |
| 5.3.1 | Model danych | 18 |
| 5.3.2 | Diagram ERD | 18 |

| | | |
|----------|--|-----------|
| 5.4 | Architektura interfejsu użytkownika | 18 |
| 5.4.1 | Projekt strony głównej | 18 |
| 5.4.2 | Projekt panelu logowania | 18 |
| 5.4.3 | Projekt mapy | 18 |
| 5.4.4 | Projekt chatu | 18 |
| 5.4.5 | Projekt forum | 18 |
| 5.4.6 | Projekt konta użytkownika | 18 |
| 6 | Przebieg realizacji projektu | 19 |
| 6.1 | Sprint 1 | 19 |
| 6.2 | Sprint 2 | 19 |
| 7 | Realizacja Projektu | 20 |
| 7.1 | Implementacja backendu | 20 |
| 7.1.1 | Struktura projektu | 20 |
| 7.1.2 | Endpointy systemu | 20 |
| 7.1.3 | Integracja z bazą danych | 23 |
| 7.1.4 | Obsługa uwierzytelnienia | 23 |
| 7.1.5 | Konteneryzacja | 23 |
| 7.2 | Implementacja frontendu | 23 |
| 7.2.1 | Struktura aplikacji | 23 |
| 7.2.2 | Zarządzanie stanem i przepływ danych | 28 |
| 7.2.3 | Integracja i komunikacja z backendem | 28 |
| 7.2.4 | Style | 28 |
| 7.2.5 | Strona główna | 28 |
| 7.2.6 | Mapa | 28 |
| 7.2.7 | Chat | 28 |
| 7.2.8 | Forum | 28 |
| 7.2.9 | Konto użytkownika | 28 |
| 7.2.10 | Panel logowania | 28 |
| 7.3 | Implementacja CI/CD | 28 |

| | |
|---|-----------|
| 8 Testy | 29 |
| 8.1 Testy jednostkowe | 29 |
| 8.2 Testy integracyjne | 29 |
| 8.3 Testy E2E | 29 |
| 8.4 Wyniki testów i wnioski | 29 |
| 9 Prezentacja systemu | 30 |
| 9.1 Strona główna | 30 |
| 9.2 Strona mapy | 30 |
| 9.3 Strona chatu | 30 |
| 9.4 Strona forum | 30 |
| 9.5 Panel logowania | 30 |
| 9.6 Panel konta użytkownika | 30 |
| 10 Nakład pracy | 31 |
| 10.1 Ogólny nakład pracy | 31 |
| 10.2 Indywidualne nakłady pracy | 31 |
| 10.2.1 Adam Langmesser | 31 |
| 10.2.2 Mateusz Redosz | 31 |
| 10.2.3 Stanisław Oziemczuk | 34 |
| 10.2.4 Kacper Badek | 34 |
| 11 Podsumowanie | 35 |
| 11.1 Osiągnięte rezultaty | 35 |
| 11.2 Napotkane wyzwania | 35 |
| 11.3 Plany na przyszłość | 35 |
| 12 Słownik pojęć i skrótów | 36 |
| Załączniki | 39 |

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

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 [DAD]** w wariancie **Lean Life Cycle [DAD-LLF]**, ponieważ podejście to łączy pożądane cechy Agile i Waterfall, a jednocześnie eliminuje stałe sprinty na rzecz pracy w ciągłym przepływie.

Kluczowe argumenty wyboru:

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

3.1.4 Narzędzia i komunikacja

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

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

3.1.5 Podział ról w zespole

- Adam
- Stanisław
- Kacper
- Mateusz

Każdy z członków zespołu uczestniczy również w przygotowaniu dokumentacji.

3.2 Harmonogram projektu

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

Rok 2024

Czerwiec • Zebranie zespołu.

- Rozważenie potencjalnych pomysłów.

Lipiec • Wybór technologii.

- Wstępne założenia architektoniczne.

Sierpień • *(do uzupełnienia)*

- *(do uzupełnienia)*

Wrzesień • *(do uzupełnienia)*

- *(do uzupełnienia)*

Październik • *(do uzupełnienia)*

- *(do uzupełnienia)*

Listopad • (do uzupełnienia)

- (do uzupełnienia)

Grudzień • (do uzupełnienia)

- (do uzupełnienia)

Rok 2025

Styczeń • (do uzupełnienia)

- (do uzupełnienia)

Luty • (do uzupełnienia)

- (do uzupełnienia)

Marzec • (do uzupełnienia)

- (do uzupełnienia)

Kwiecień • (do uzupełnienia)

- (do uzupełnienia)

Maj • (do uzupełnienia)

- (do uzupełnienia)

Czerwiec • (do uzupełnienia)

- (do uzupełnienia)

Lipiec • (do uzupełnienia)

- (do uzupełnienia)

Sierpień • (do uzupełnienia)

- (do uzupełnienia)

Wrzesień • (do uzupełnienia)

- (do uzupełnienia)

Październik • (*do uzupełnienia*)

- (*do uzupełnienia*)

Listopad • (*do uzupełnienia*)

- (*do uzupełnienia*)

Grudzień • (*do uzupełnienia*)

- (*do uzupełnienia*)

Rok 2026

Styczeń • (*do uzupełnienia*)

- (*do uzupełnienia*)

3.3 Technologie i narzędzia

Do realizacji projektu wykorzystano wiele technologii oraz narzędzi informatycznych. Przy wyborze technologii kierowaliśmy się ich popularnością, dostępnością dokumentacji oraz artykułów, a także łatwością użycia. Ponadto staraliśmy się korzystać z rozwiązań, które poznaliśmy w trakcie studiów. Narzędzia zostały dopasowane do wybranych technologii. Poniżej przedstawiono opis wybranych opcji.

3.3.1 Technologie

3.3.2 Narzędzia

Do niektórych płatnych narzędzi mieliśmy bezpłatny dostęp za pośrednictwem uczelni, w innych mogliśmy założyć konta edukacyjne, które oferowały dostęp do wszystkich funkcji narzędzia. Gdy żadna z wymienionych opcji nie była udostępniona, wybieraliśmy rozwiązania darmowe.

- IntelliJ IDEA Ultimate

Jest to IDE od firmy JetBrains. Dzięki wielu dostępnym pluginom oferuje obsługę wielu języków programowania oraz innych składni. Pozwala również na integrację z repozytorium. Używamy 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. Wykorzystujemy je do lokalnego uruchamiania bazy danych oraz serwisów do cachowania.

- **One Drive**

Usługa dysku chmurowego oferowana przez firmę Microsoft. Przechowujemy 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 API REST usługi Azure Storage. Wykorzystaliśmy go do przechowywania zdjęć profilowych użytkownika oraz multimedii (zdjęcia i filmy) ze spotów i forum.

- **Jira**

To narzędzie firmy Atlassian do zarządzania pracami nad projektem w metodach zwinnych. Do Backlogu wpisywaliśmy zadania, a na Tablicy Kanbanowej rejestrowaliśmy ich statusy oraz poświęcony czas.

- **GitHub**

Zdalne repozytorium służące do przechowywania i wersjonowania kodu aplikacji. Zamieściliśmy tam kod naszego projektu. Do każdego zadania tworzyliśmy osobną gałąź z właściwą nazwą, a po zakończeniu prac przeprowadzaliśmy Review kodu. Następnie łączymy ją do głównej gałęzi deweloperskiej.

- **GitHub Actions**

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

- **GitHub Copilot**

To narzędzie sztucznej inteligencji będące asystentem programisty. W projekcie analizuje plik oraz pliki powiązane. Wykorzystywaliśmy go podczas Review kodu. Copilot skanował wszystkie pliki i w komentarzach opisywał sugerowane zmiany lub potencjalne błędy.

- **Discord**

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

- **Messenger**

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

- **Postman**

To narzędzie służące do testowania endpointów API. Pozwala grupować pytania w kolekcje, wysyłać ich różne typy oraz analizować odpowiedzi z serwera. Wykorzystywaliśmy go do testowania stworzonych endpointów oraz debugowania.

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

4.3.5 Funkcjonalności dla logowania i rejestracji

4.3.6 Funkcjonalności dla wyszukiwarki spotów

4.3.7 Funkcjonalności dla motywu

4.4 Wymagania pozafunkcjonalne

4.5 Wymagania interfejs z otoczeniem

4.6 Wymagania na środowisko docelowe¹⁷

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 UserNotFoundException.

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

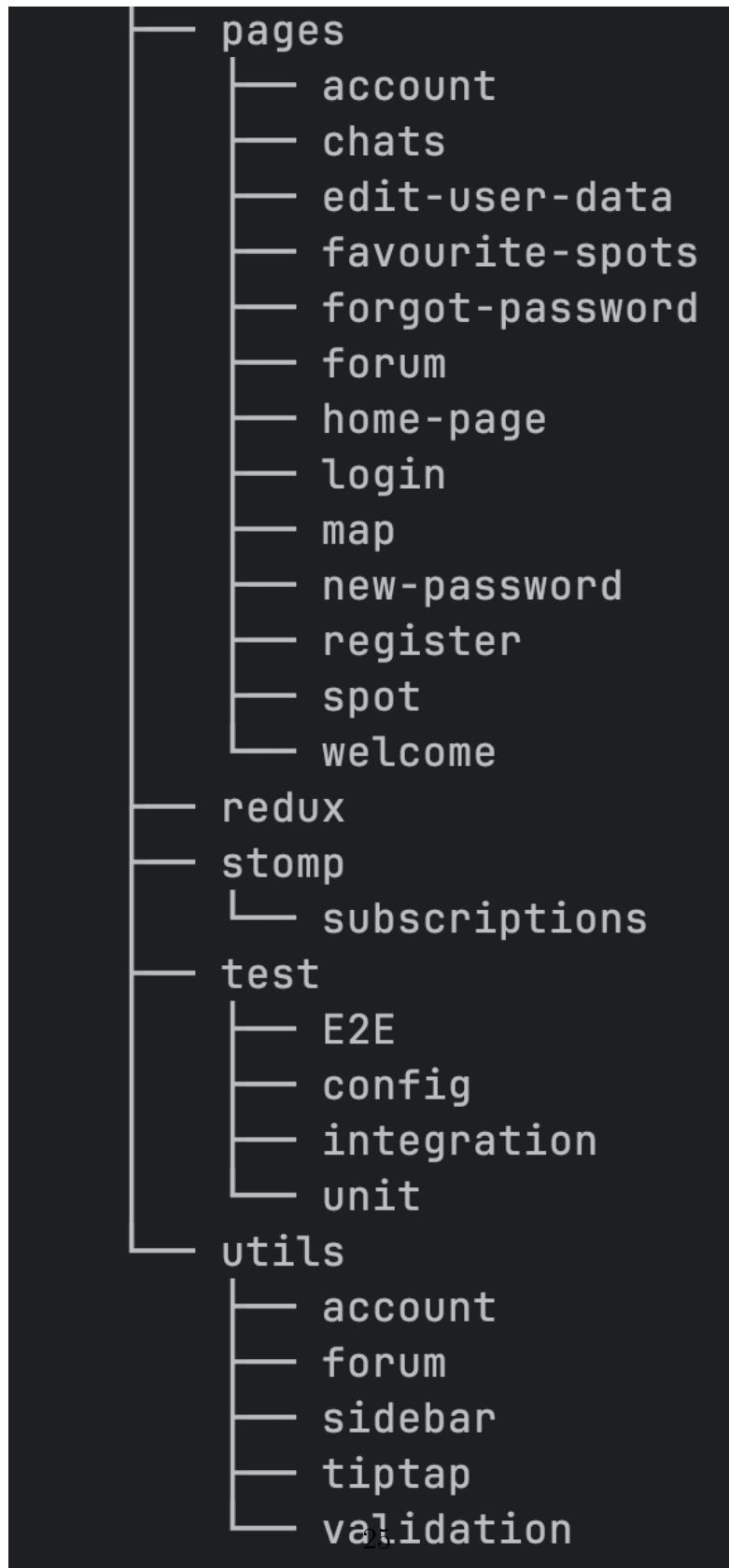
7.2 Implementacja frontendu

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 jest przedstawione na rysunkach 7.1 oraz 7.2.



Rysunek 7.1: Struktura katalogów (1)



Rysunek 7.2: Struktura katalogów (2)

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

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

Rysunek 7.3: Implementacja routera (1)

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

export default router;
```

Rysunek 7.4: Implementacja routera (2)

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

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

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

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

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

Rysunek 7.5: Implementacja komponentu bramki (`ProtectedRoute`)

7.2.2 Zarządzanie stanem i przepływ danych

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 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 Sidebaru
- Zmiana kropki na przyciemnienie tła na Sidebar
- Poprawa strony do logowania i rejestracji
- Usunięcie username z account Redux
- Dodanie zamknięcia Sidebaru 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

Backend Część aplikacji odpowiedzialna za logikę biznesową, przetwarzanie danych i komunikację z bazą danych. Działa po stronie serwera i obsługuje żądania wysyłane przez frontend. 2, 13, 32

Backlog Lista zadań, które należy wykonać w ramach projektu, używane w metodykach zwinnych.. 13

Biblioteka Zewnętrzny lub wewnętrzny zestaw gotowych funkcji, klas, komponentów lub modułów, który można wielokrotnie wykorzystywać w projekcie zamiast pisać wszystko od zera. 26

CI/CD Skrót od *Continuous Integration/Continuous Deployment*. Praktyka programistyczna polegająca na automatyzacji procesu budowania, testowania i wdrażania oprogramowania. 14, 31, 34

Design Etap lub proces projektowania wyglądu i funkcjonalności aplikacji, obejmujący zarówno aspekty wizualne, jak i użytkowe (UX/UI). 31

Disciplined Agile Delivery - Lean Life Cycle Kanbanowy tryb pracy bez sprintów: zespół pobiera małe zadania z tablicy, pilnuje limitów pracy w toku i często wdraża małe zmiany. Planowanie jest lekkie i na bieżąco; priorytet mają rzeczy o najwyższej wartości. Skupiamy się na płynności—skracamy się

czas przejścia zadań, usuwa blokady, usprawnia proces małymi krokami. Jakość jest wbudowana: code review, testy automatyczne i prosta definicja „gotowe”. 8

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. 23

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, 13, 32

IDE ang. integrated development environment, to zintegrowane środowisko programistyczne, służące do tworzenia, modyfikowania, testowania i konserwacji oprogramowania. 13

JWT Skrót od *JSON Web Token*. Standard służący do bezpiecznego przekazywania informacji między stronami w formacie JSON, często używany w procesach autoryzacji użytkowników. 31

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 przekierowany (np. na stronę główną). 27

Review kodu Proces polegający na wzajemnym przeglądzie kodu źródłowego przez programistów w celu wykrycia błędów, poprawy jakości oraz zwiększenia spójności projektu. 13, 14, 31

Sidebar Boczny panel w interfejsie użytkownika, zawierający menu nawigacyjne lub dodatkowe opcje funkcjonalne aplikacji. 31–33

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 „To do” lub „In progress”.. 13

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.