

**Wyniki etapu I: Wizja, słownik,
model domenowy, reguły biznesowe**

„Turysta Z Odznaką”

Projektowanie oprogramowania

Skład zespołu:
Klaudia Błażyczek
Justyna Małuszyńska

Prowadzący:
Dr.Inż. Zbigniew Szpunar

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

Wizja i słownik

1. Wprowadzenie

Górska Odznaka Turystyczna PTTK (w skrócie GOT PTTK) została ustanowiona i wprowadzona w życie przez Polskie Towarzystwo Tatrzańskie celem promocji turystyki górskiej pośród coraz większej ilości osób. Nadzór nad odznaką sprawuje Komisja Turystyki Górskiej Zarządu Głównego PTTK.

GOT ma następujące kategorie i stopnie:

- „W góry”:
 - Brązowa
 - Srebrna
 - Złota
- Popularna
- Mała:
 - Brązowa
 - Srebrna
 - Złota
- Duża:
 - Brązowa
 - Srebrna
 - Złota
- „Za wytrwałość”:
 - Mała
 - Duża

Zdobywanie kolejnych rodzajów odznaki możliwe jest dzięki odbywaniu wycieczek po wybranych trasach górskich. Wycieczki te mogą odbywać się przez cały rok kalendarzowy, mając na uwadze odpowiednie warunki pogodowe jak i terenowe.

Postępy w zdobywaniu odznak odnotowywane są w samodzielnie prowadzonej ksiązeczce, w której zapisywany jest dokładny przebieg przebytych tras wraz z wyszczególnieniem konkretnych odcinków oraz należnych za nie punktów. Punkty przydzielane są na podstawie przygotowanej przez Komisję Turystyki Górskiej punktacji, dostępnej dla turysty na stronie WWW w „Trasach punktowanych do GOT PTTK”. W przypadku, gdy trasa nie jest uwzględniona w trasach punktowanych, obowiązuje następujący sposób zliczania punktów:

- 1 przebyty kilometr = 1 punkt (zaokrąglenie w górę do jednego punktu przy przebyciu co najmniej 500 m)
- 100 m pokonanej sumy różnic poziomów przy podejściach = 1 punkt (zaokrąglenie w górę do jednego punktu przy minimum 51 m sumy podejścia)

Turysta, planując dzienne wycieczki, musi pamiętać, że obowiązuje maksymalna norma zliczana do odznaki na dziennej trasie w wysokości 50 punktów, wyjątek stanowi GOT „W góry” – 15 punktów. Nie wolno także zaliczać punktów uzyskanych na tych samych odcinkach i w tym samym kierunku w toku zdobywania tego samego stopnia GOT PTTK. Mimo, iż czas zdobywania poszczególnych rodzajów i stopni odznaki jest nieograniczony, należy mieć na uwadze, że w ciągu jednego roku kalendarzowego można zdobyć tylko jeden stopień GOT.

GOT PTTK popularna i mała jest możliwa do zdobycia po ukończeniu 7. roku życia. Obowiązują następujące progi punktowe dla każdego ze stopni:

- Popularna – 60 punktów;
- Mała brązowa – 120 punktów;
- Mała srebrna – 360 punktów;
- Mała złota – 720 punktów;

Dla osób niepełnosprawnych obowiązują powyższe progi punktowe obniżone o 25%, a więc kolejno: 45, 90, 270 oraz 540 punktów.

GOT w stopniu popularnym i małym brązowym mogą być zdobyte w jednym roku

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

kalendarzowym pod warunkiem uzyskania co najmniej 180 punktów (135 punktów w przypadku osób niepełnosprawnych). Przy weryfikacjach odznak w stopniach popularnym, małym brązowym lub małym srebrnym można na poczet następnego małego stopnia odznaki zaliczyć nadwyżkę uzyskanych punktów do wysokości 50% normy potrzebnej do jego zdobycia.

Do weryfikacji zdobytych stopni Górskiej Odznaki Turystycznej upoważnieni są przodownicy turystyki górskiej PTTK. Jest on także uprawniony do potwierdzania odbycia wycieczek:

- a. W zakresie posiadanych uprawnień na poszczególne grupy górskie, nawet jeśli nie był obecny na wycieczce,
- b. Poza zakresem swych uprawnień, jeśli brał w nich udział.

Potwierdzić odbycie zaplanowanej wycieczki może także przewodnik górski, wyłącznie w przypadku, gdy brał udział w tej wycieczce.

Zebranie tych wszystkich informacji i reguł w jednym miejscu umożliwi szybsze i wygodniejsze planowanie wycieczek górskich oraz przydzielanie należnych punktów za ich przebycie dla GOT popularnej i małej (z przestrzeganiem obowiązujących reguł i zasad). Usprawniony zostanie także proces weryfikacji zdobytych odznak, dzięki przetrzymywaniu dokumentacji potwierdzającej przebycie danych odcinków tras (potwierdzenie przez przodownika lub przewodnika, fotografia z datą, zapis położenia z wykorzystaniem GPS). Użytkownicy będą mogli swobodnie wyszukiwać, grupować, filtrować a następnie wyświetlać wybrane odcinki tras – wyłącznie z eksportem do formatu PDF. System umożliwi przechowywanie własnych odcinków dodanych przez użytkownika, wraz z archiwizacją odbytych wycieczek. Ponadto system będzie monitorował dostępność odcinków i powiadamiał użytkowników o czasowo zamkniętych szlakach lub całkowitym wyłączeniu ich z użytku.

System nie będzie wspierał zdobywania i weryfikacji pozostałych GOT, tj. „W góry”, duża, „Za wytrwałość”. Ponadto nie będzie umożliwiał udostępniania zaplanowanych wycieczek pomiędzy użytkownikami systemu ani automatycznego planowania wycieczek poprzez podanie punktów początkowych, końcowych i (ewentualnie) pośrednich.

2. Pozycjonowanie

2.1 Sformułowanie problemu

Problem	Trudny dostęp dla użytkowników do tras punktowanych do GOT oraz ich sposobu punktacji co generuje problem podczas ustalania trasy wycieczki oraz obliczania należnych za nie punktów zgodnie z obowiązującym regulaminem. Występujący problem ze zdobyciem odpowiednich pieczętek potwierdzających odbycie wycieczki po dotarciu na miejsce.
Dotyczy	Turyści zdobywający Górską Odznakę Turystyczną PTTK.
Wpływ problemu	Turyści zmuszeni do ręcznego wyszukiwania tras oraz ich punktacji na przestarzałej stronie internetowej, co przekłada się na dłuższy czas planowania wycieczek. Utrudnienie zdobycia dowodów odbycia tras potrzebnych do weryfikacji odznak. Wymóg ręcznego przeliczania punktów za trasę.
Pomyślne rozwiązanie	Ułatwiony dostęp do spisu odcinków tras górskich oraz ich punktacji w dowolnym miejscu i czasie. Ułatwienie prowadzenia dokumentacji odbytych wycieczek. Łatwy i intuicyjny sposób układania tras wycieczkowych.

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

	Automatyczne przeliczanie punktów za trasy. Przyspieszenie procesu weryfikacji odznak.
--	---

2.2 Opis pozycji produktu

Dla	Komisja Turystyki Górskiej ZG PTTK
Którzy	Chcą ułatwić turystom organizowanie wycieczek górskich oraz zdobywanie GOT PTTK.
„Turysta Z Odznaką”	Aplikacja webowa/desktopowa.
Który	W łatwy sposób umożliwia wyszukiwanie, planowanie oraz dokumentowanie przebycia punktowanych tras górskich Górskiej Odznaki Turystycznej. Automatycznie przelicza zdobyte punkty na podstawie przebytych tras wycieczkowych. Upraszczając proces weryfikacji – więcej możliwości dokumentowania odbytych wycieczek.
Inaczej niż	Strona internetowa Komisji Turystyki Górskiej PTTK oraz obecny system weryfikacji z wykorzystaniem książeczek.
Nasz produkt	Umożliwia łatwe wyszukiwanie tras, planowanie wycieczek, dokumentowanie przebycia określonych tras w celu zdobywania Górskiej Odznaki Turystycznej. Wspiera proces przeliczania punktów i weryfikacji odznak. Przechowuje zapisane trasy wraz z historią.

3. Opis udziałowców i użytkowników

3.1 Podsumowanie udziałowców

Nazwa	Opis	Zakres odpowiedzialności
Administrator systemu	Osoba odpowiedzialna za prawidłowe działanie systemu.	Nadawanie uprawnień użytkownikom systemu. Rozwiązywanie bieżących problemów z działaniem systemu.
Członek Komisji Turystyki Górskiej	Osoba należąca do Komisji Turystyki Górskiej Zarządu Głównego PTTK	Pomaganie w tworzeniu oprogramowania pod wymagania PTTK.
Wykonawca zlecenia	Wykonawca zleceń: Justyna i Klaudia	Osoby odpowiedzialne za stworzenie dokumentacji i wdrożenie systemu.

3.2 Podsumowanie użytkowników

Nazwa	Opis	Zakres odpowiedzialności
Turysta	Osoba zidentyfikowana, wędrująca po szlakach górskich.	Wyszukiwanie informacji o odcinkach tras oraz ich punktacji, dodawanie własnych odcinków, planowanie tras, przeglądanie odznak i dokumentacji

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

Nazwa	Opis	Zakres odpowiedzialności
Przewodnik górski	Osoba zawodowo oprowadzająca turystów po górach.	<i>Wyszukiwanie informacji o odcinkach oraz ich punktacji, planowanie tras, zatwierdzanie przebycia odcinków tras przez turystów w ramach zdobywania GOT, dodawanie własnych odcinków, logowanie/rejestracja w systemie, przeglądanie odznak i dokumentacji, zgłoszenie Członkom Komisji Turystyki Górskiej o potrzebie chwilowego zamknięcia odcinka górskego.</i>
Przodownik turystyki górskiej	Osoba popularyzująca turystykę górską i wiedzę o górach.	<i>Wyszukiwanie informacji o odcinkach tras oraz ich punktacji, planowanie tras, zatwierdzanie i weryfikacja dowodów przebycia odcinków tras przez turystów w ramach zdobywania GOT, dodawanie własnych odcinków, logowanie/rejestracja w systemie, przeglądanie odznak i dokumentacji, zgłoszenie Członkom Komisji Turystyki Górskiej o potrzebie chwilowego zamknięcia odcinka górskego.</i>
Członek Komisji Turystyki Górskiej	Osoba należąca do Komisji Turystyki Górskiej Zarządu Głównego PTTK	<i>Wyszukiwanie informacji o odcinkach tras oraz ich punktacji, planowanie tras, dodawanie własnych odcinków, logowanie/rejestracja w systemie, przeglądanie odznak i dokumentacji, zarządzanie odcinkami punktowanymi tras, zarządzanie punktami opisanymi GOT.</i>
Gość	Osoba bliżej nieznana	<i>Wyszukiwanie informacji o odcinkach tras oraz ich punktacji, logowanie/rejestracja w systemie</i>

4. Opis produktu

4.1 Potrzeby i cechy

Potrzeba	Priorytet	Cechy	Planowane wydanie
System logowania i rejestracji użytkowników	Must	Każdy użytkownik posiada możliwość rejestracji do systemu. Po rejestracji użytkownik jest użytkownikiem identyfikowanym w bazie, posiada swoje uprawnienia i może logować się do systemu.	1.0
Dodawanie, usuwanie i modyfikacja odcinków oraz punktów geograficznych	Must	Dodawanie, usuwanie i modyfikacja rekordów w bazie dotyczących punktowanych odcinków oraz punktów geograficznych.	1.0
Filtrowanie danych o odcinkach	Should	Filtrowanie odcinków według ustalonych kryteriów (obszar, liczba punktów, kolor szlaku, dostępność, kierunkowość, długość, różnice wysokości punktów, atrybut „przez”).	1.1

“Turysta Z Odznaką”<Project Name>			
Etap I			Data: <dd/mmm/yy>
Wyświetlanie tras wraz z należną punktacją	Must	Wyświetlenie w czytelny sposób wyszukanych tras oraz przypisanej do nich punktacji możliwej do zdobycia.	1.0
Planowanie wycieczek górskich	Must	Skomponowanie z użyciem odcinków punktowanych i własnych wycieczki górskiej. Wyświetlenie danych do skomponowanej trasy (punkty, kolory szlaków, kierunki odcinków, atrybut „przez”).	1.0
Dodawanie nowych, niepunktowanych odcinków tras	Should	Dodawanie nowych rekordów w bazie dotyczących własnych odcinków tras, poprzez podanie punktów: początkowego i końcowego.	1.1
System monitorujący dostępność odcinków górskich	Could	Status dostępności odcinków monitorowany w czasie rzeczywistym (dodawanie, usuwanie, modyfikacja odcinków). Wyświetlanie dostępności odcinków (otwarty, czasowo zamknięty – od kiedy do kiedy, zlikwidowany na stałe).	Finalne
System eksportowania zapisanych tras	Could	Weksportowanie zapisanych tras wycieczek do formatu PDF.	Finalne
System archiwizujący nieistniejące odcinki tras	Should	Przechowywanie w bazie nieistniejących już (zlikwidowanych) odcinków górskich. Wyświetlanie informacji na temat zlikwidowanych odcinków górskich.	1.1
Automatyczne zliczanie punktów i odznak	Must	Automatyczne zliczanie punktów zgodnie z zasadami regulaminowymi, po weryfikacji odbytych odcinków wycieczki. Automatyczne przyznanie odznaki po zdobyciu odpowiedniej ilości punktów (określonej w regulaminie).	1.0
Przeglądanie odznak wraz z dokumentacją	Must	Dostęp do zdobytych odznak wraz z historią przebytych tras oraz dokumentacją.	1.0
Wysyłanie prośby o zatwierdzenie lub weryfikację przebycia odcinków tras w ramach wycieczki	Should	Wysłanie prośby o zatwierdzenie lub weryfikację przebycia zaplanowanych odcinków trasy do przodownika lub przewodnika.	1.0
System wspierający zatwierdzanie odbytych tras	Should	Wyświetlanie przodownikom turystyki górskiej PTTK oraz przewodnikom górskim prośb o potwierdzenie ostatnio zdobytych punktów odcinków. Zatwierdzenie lub odrzucenie prośby o weryfikację zdobycia punktów. Zatwierdzenie lub odrzucenie innych dowodów odbycia tras (fotografia, zapis położenia GPS).	1.0

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

Zgłaszanie informacji o awarii danego odcinka trasy	Could	Zgłaszczenie Członkom Komisji o aktualnie zamkniętych odcinkach ze względu na występujące na nich czasowe utrudnienia.	Finalne
Załączanie dokumentacji do odznak (dowody przebycia trasy)	Must	Dodłaczanie do zaplanowanej i odbytej wycieczki dowodów ich przebycia w postaci pliku (fotografia punktu początkowego i końcowego z datą) lub zapis położenia GPS	1.0

5. Inne wymagania produktowe

Wymaganie	Priorytet	Planowane wydanie
Uruchomienie systemu łatwego do użycia, intuicyjność - Użytkownicy nie zgłaszają większych zastrzeżeń co do intuicyjności interfejsu.	Must	1.0
Stworzenie solidnego systemu - System obsługuje ruch przynajmniej 10.000 użytkowników.	Must	Finalne
Aplikacja dostępna 24/7.	Should	1.1
Bezpieczeństwo - System jest zabezpieczony przed hakerami i złośliwym oprogramowaniem w przynajmniej 95% ataków.	Must	1.0
Wydajność - Użytkownik czeka max. 5s na odpowiedź systemu.	Should	Finalne
Dostępność - System działa we wszystkich przeglądarkach.	Must	1.1
Niezawodność - W przypadku awarii systemu nie następuje usunięcie danych.	Must	1.0

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

6. Słownik pojęć

Termin	Synonimy	Definicja terminu
Członek Komisji	Członek Komisji Turystyki Górskiej, Członek zarządu	Osoba należąca do Komisji Turystyki Górskiej Zarządu Głównego PTTK. Jest odpowiedzialny za modyfikowanie i usuwanie danych o odcinkach tras punktowanych do GOT.
Dowód przebycia odcinka	Potwierdzenie przebycia odcinka, Dowód	Dowód przebycia odcinka jest potwierdzeniem przebycia danego odcinka (niezbędnym do uzyskania punktów) przez turystę. Dowodem może być: <ul style="list-style-type: none"> - Elektroniczny podpis przodownika - Elektroniczny podpis przewodnika - Fotografie punktów początkowych i końcowych odcinka - Pliki z zarejestrowaną przez GPS aktywnością na danych odcinkach tras
Gość		Użytkownik korzystający z aplikacji w celu wyszukiwania informacji o odcinkach tras oraz ich punktacji. Nie posiada konta w systemie.
Grupa górska	Łańcuch górski	Teren górski o równolegle ułożonych pasmach wraz z kotlinami i przylegającymi do nich pogórzami. Jest identyfikowana przez nazwę i kraj w którym leży.
Kategoria odznaki	Rodzaj	Odznaki mają dwie kategorie: popularna oraz mala. GOT PTTK w stopniach popularnym i małym brązowym można zdobyć w jednym roku kalendarzowym pod warunkiem uzyskania co najmniej 180 (niepełnosprawni 135) punktów. Przy weryfikacjach odznak w stopniach popularnym, małym brązowym lub małym srebrnym GOT PTTK można na poczet następnego malego stopnia odznaki zaliczyć nadwyżkę uzyskanych punktów do wysokości 50% normy potrzebnej do jego zdobycia.
Kolor szlaku		Oznakowanie szlaków i tras górskich, pomagające określić atrakcyjność szlaku i długość.
Kraj		Kraj określa przynależność grupy górskiej do danego państwa.
Odcinek		Droga, jaką należy przebyć, od punktu startowego do punktu końcowego. Każdy odcinek ma przypisaną odpowiednią ilość punktów, które należą się turystie za przebycie tego odcinka. Odcinek może być jednokierunkowy lub dwukierunkowy.
Odcinek nieczynny	Odcinek nieczynny trasy punktowanej GOT, Odcinek nieczynny trasy,	Do grupy odcinków nieczynnych należą te odcinki tras punktowanych GOT, które chwilowo zostały wyłączone z użytku. Powodem wyłączenia trasy z użytku mogą być przykładowo katastrofa naturalna lub prace remontowe. Proszę o uznanie trasy za nieczynną mogą złożyć przewodnik lub przodownik, zatwierdzić lub odrzucić może zaś Członek Komisji.

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

	odcinek zamknięty, odcinek zamknięty trasy punktowanej GOT, odcinek zamknięty trasy	
Odcinek opisany	Odcinek trasy punktowanej GOT	Odcinek opisany to odcinek zdefiniowany i wprowadzony do systemu przez Członka Komisji. Ilość należytych punków do GOT za dany odcinek opisany jest definiowana przez Członka Komisji.
Odcinek trasy		Odcinek wybrany przez użytkownika z bazy odcinków wchodzący w skład zaplanowanej trasy.
Odcinek własny		Odcinek jest definiowany przez turystę, z zachowaniem zasad dotyczących punktacji regulaminu górskiej odznaki turystycznej PTTK. Odcinek własny jest dostępny tylko dla twórcy tego odcinka. Odcinek własny może być jednokierunkowy lub dwukierunkowy.
Odcinek zlikwidowany	Odcinek zlikwidowany trasy punktowanej GOT, Odcinek zlikwidowany trasy	Do grupy odcinków zlikwidowanych należą te odcinki tras punktowanych GOT, które na zawsze zostały wyłączone z użytku. Odcinek może zostać zaliczony do odcinków zlikwidowanych przez Członka Komisji.
Odznaka GOT		Odznaka promująca turystykę górską.
Odznaka mała		Kategoria odznaki GOT posiadająca 3 stopnie. Ustala się następujące ilości punktów koniecznych do zdobycia poszczególnych stopni małych GOT PTTK: <ul style="list-style-type: none"> - brązowy 120 punktów, - srebrny 360 punktów, - złoty 720 punktów. <p>Turyści niepełnosprawni mogą skorzystać przy ich zdobywaniu z norm pomniejszonych o 25%, tj. odpowiednio 90, 270 i 540 punktów.</p>
Odznaka popularna		Kategoria odznaki GOT posiadająca jeden stopień. Ustala się następującą ilość punktów koniecznych do zdobycia GOT PTTK: <ul style="list-style-type: none"> - popularny 60 punktów. <p>Turyści niepełnosprawni mogą skorzystać przy ich zdobywaniu z norm pomniejszonych o 25%, tj. odpowiednio 45 punktów.</p>

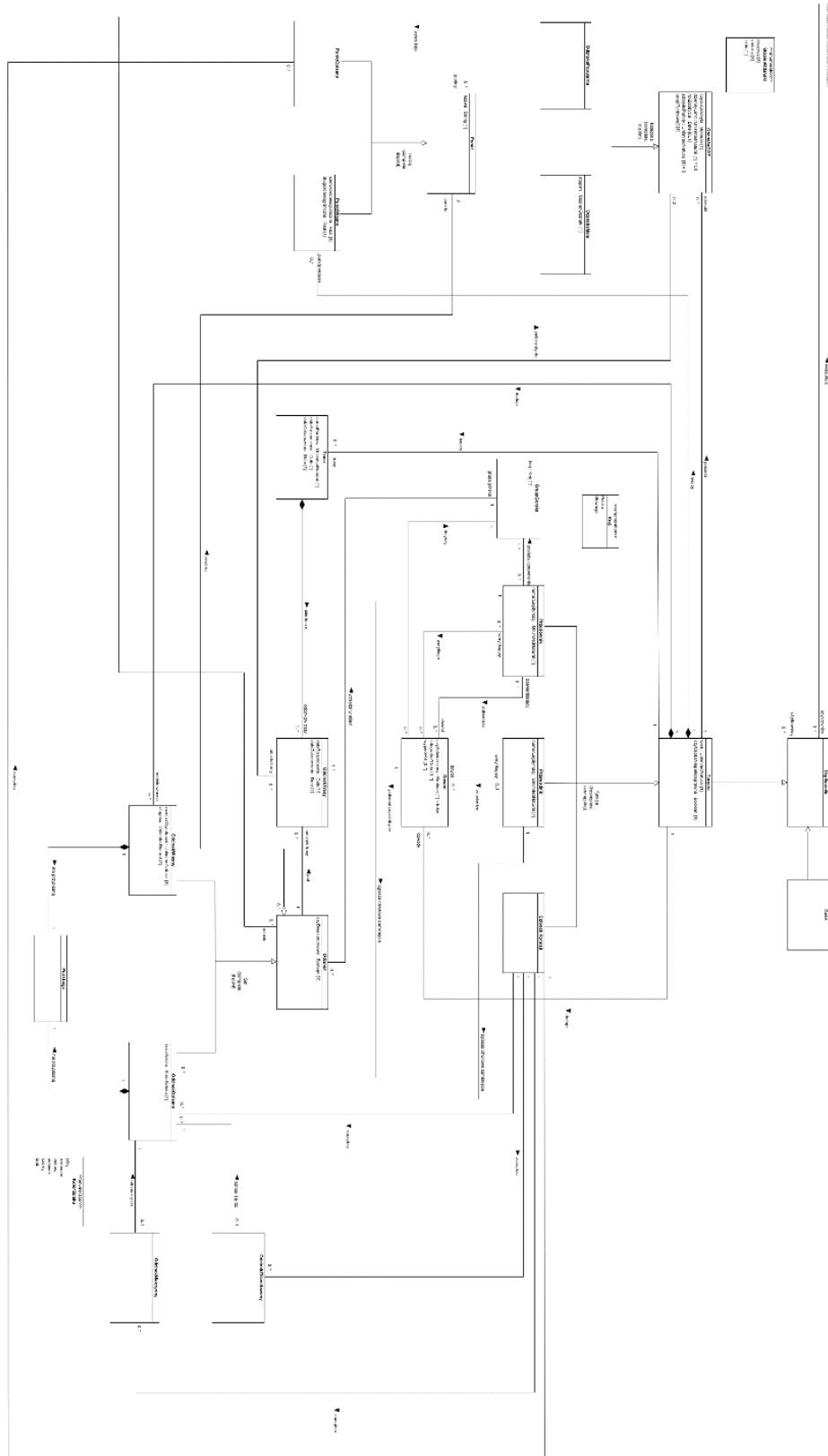
“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

Pasmo górskie		Samodzielna grupa górska o wyraźnie wydłużonym kształcie, mająca grań główną (często wododziałową). W większości przypadków jest ono częścią łańcucha górskiego, np. Karkonosze.
Przewodnik	Pilot, opiekun, przewodnik górski, przewodnik turystyki górskiej, przewodnik turystyki górskiej PTTK	Osoba zawodowo oprowadzająca turystów po odcinkach i trasach górskich. Udziela o nich fachowej informacji oraz sprawuje opiekę nad turystami. Posiada legitymację i odznakę potwierdzającą jego uprawnienia. Jest uprawniony do potwierdzania przebycia przez podróżników odcinków i tras górskich, po których ich oprowadzał, w ramach zdobywania GOT PTTK.
Przodownik	Przodownik turystyki górskiej, przodownik turystyki górskiej PTTK	Osoba zawodowo oprowadzająca turystów po odcinkach i trasach górskich. Udziela o nich fachowej informacji oraz sprawuje opiekę nad turystami. Posiada legitymację i odznakę potwierdzającą jego uprawnienia. Jest uprawniony do potwierdzania odbycia wycieczek w zakresie posiadanych uprawnień na poszczególne grupy górskie, nawet jeśli nie był obecny na wycieczce, a poza zakresem swoich uprawnień jeśli brał w niej udział. Weryfikuje dowody przebycia odcinków tras (fotografia, zapis GPS).
Punkt	Miejsce, punkt geograficzny	Okręślone miejsce w terenie. Rozróżnia się dwa rodzaje punktów: punkt własny oraz punkt opisany.
Punkt opisany		Punkt identyfikowany przez nazwę, wchodzi w skład odcinków punktowanych do GOT. Jest definiowany przez Członka Komisji.
Punkt własny		Punkt identyfikowany przez nazwę i współrzędne geograficzne. Punkt własny jest widzialny tylko dla turysty, który go stworzył.
Punktacja		Ilość punktów przyznawana za pokonanie danego odcinka wycieczki górskiej. W przypadku odcinków punktowanych do GOT, stosować należy podaną z góry punktację. Jeżeli odcinek wycieczki nie jest ujęta odcinkach punktowanych do GOT PTTK, to przyznaje się jeden punkt za każdy przebyty kilometr oraz jeden punkt za każde pokonane 100 m sumy różnic poziomów przy podejściach (zaokrąglenie w górę do jednego punktu przy minimum 51 m sumy podejść lub ponad 500 m przebytej trasy).
Stopień odznaki	Poziom, ranga	Stopień odznaki to szczebel w hierarchii kategorii odznak. Kategoria odznaki popularna ma tylko jeden stopień. Kategoria odznaki mała ma trzy stopnie: brązowy, srebrny i złoty. W jednym roku kalendarzowym można zdobyć tylko jeden stopień GOT PTTK, z wyjątkiem GOT PTTK popularnej i malej brązowej, które mogą być zdobyte w tym samym roku.
Trasa	Wycieczka	Trasa składa się z przynajmniej jednego odcinka trasy. Odcinki trasy składające się na trasę muszą tworzyć ciągłość, to znaczy, w miejscu zakończenia jednego odcinka trasy, powinien zaczynać się nowy odcinek trasy. W skład tras mogą wchodzić odcinki własne. Zdarzyć się może również, że w skład tras wchodzić będą odcinki nieczynne i odcinki zlikwidowane – dotyczy to tras zdobytych

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

		przed zamknięciem lub likwidacją danych odcinków tras. Trasa musi mieć nazwę własną.
Turysta	Podróżnik, wędrownik, wędrowiec, piechur	Osoba uprawiająca turystykę, wędrująca po szlakach górskich, która ukończyła 7 rok życia.
Użytkownik		Osoba korzystająca z systemu. Ma możliwość zarejestrowania i zalogowania w systemie. W przypadku, gdy użytkownik nie jest zalogowany, posiada uprawnienia gościa. Po zalogowaniu może posiadać uprawnienia: turysty, przewodnika, przodownika lub Członka Komisji.

Model domenowy



“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

Reguły biznesowe

<<BusinessRule>> Wiek turysty Text = "Turysta musi mieć ukończony 7 rok życia." typ = "ograniczenie strukturalne"	<<BusinessRule>> Odcinek własny turysty Text = "Odcinek własny musi być powiązany z dokładnie jednym turystą." typ = "ograniczenie strukturalne"	<<BusinessRule>> Dowód potwierdzający odcinek trasy Text = "Dowód musi potwierdzać przebycie co najmniej jednego odcinka trasy w obrębie jednej grupy górskiej." typ = "ograniczenie strukturalne"
<<BusinessRule>> Limit dziennych punktów Text = "Maksymalna ilość punktów zdobytych jednego dnia przez turystę w ramach zdobywania odznaki, nie może przekroczyć 50 punktów." typ = "ograniczenie strukturalne"	<<BusinessRule>> Punkt własny turysty Text = "Punkt własny musi być powiązany z dokładnie jednym turystą." typ = "ograniczenie strukturalne"	<<BusinessRule>> Odcinek grupy górskiej Text = "Odcinek musi wchodzić w skład jednej grupy górskiej." typ = "ograniczenie strukturalne"
<<BusinessRule>> Identyfikacja punktu opisanego Text = "Punkt musi być identyfikowany przez swoją nazwę." typ = "ograniczenie strukturalne"	<<BusinessRule>> Status odznaki Text = "Odznaka musi mieć ustalony status." typ = "ograniczenie strukturalne"	<<BusinessRule>> Kolor szlaku Text = "Odcinki opisane mogą mieć określony kolor szlaku." typ = "ograniczenie strukturalne"
<<BusinessRule>> Identyfikacja punktu własnego Text = "Punkt własny musi być identyfikowany przez współrzędne geograficzne. Wszystkie punkty własne turysty muszą posiadać unikalną nazwę." typ = "ograniczenie strukturalne"	<<BusinessRule>> Punktacja odznaki Text = "Odcinek musi mieć przypisaną punktację." typ = "ograniczenie strukturalne"	<<BusinessRule>> Odcinek opisany Text = "Odcinek opisany musi posiadać atrybut „przez”, który domyślnie przyjmuje wartość pustego napisu."
<<BusinessRule>> Kierunkowość odcinka Text = "Odcinek może być jednokierunkowy lub dwukierunkowy." typ = "ograniczenie strukturalne"	<<BusinessRule>> Identyfikacja przodownika Text = "Przodownik musi posiadać unikalny numer legitymacji." typ = "ograniczenie strukturalne"	<<BusinessRule>> Identyfikacja przewodnika Text = "Przewodnik musi posiadać unikalny numer legitymacji." typ = "ograniczenie strukturalne"

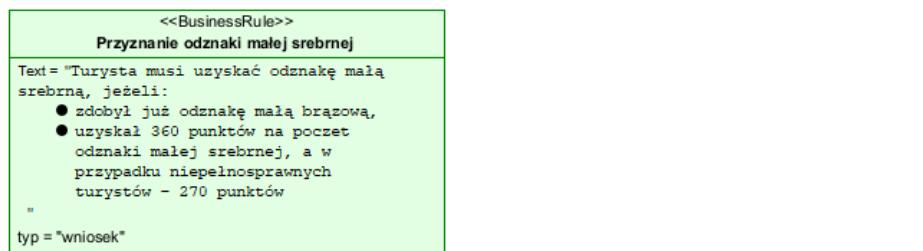
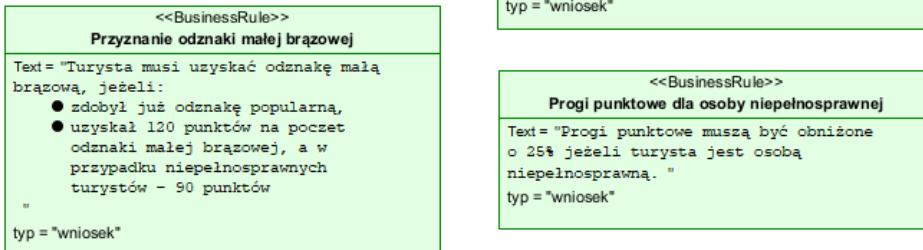
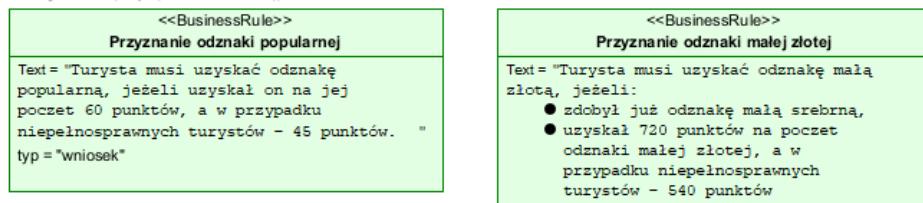
“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

Visual Paradigm Standard (Justyna/Institute of Informatics)

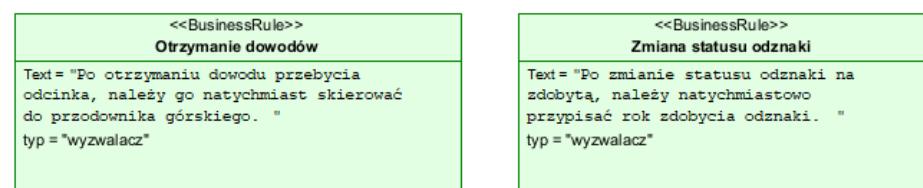
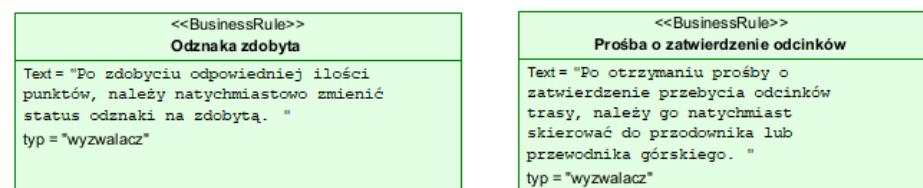
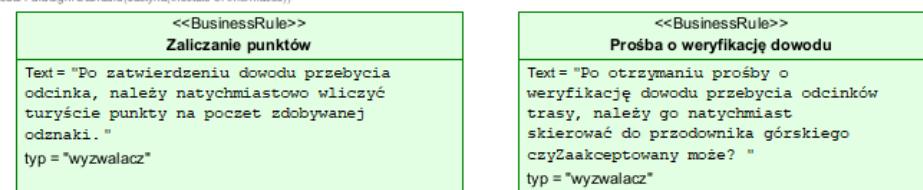
<<BusinessRule>> Niezaliczanie punktów	<<BusinessRule>> Forma dowodu	<<BusinessRule>> Odcinki opisane
Text = "Nie można zaliczać punktów uzyskanych na tych samych odcinkach i w tym samym kierunku w toku zdobywania tego samego stopnia GOT. " typ = "ograniczenie operacji"	Text = "Dowód może być przyjęty tylko w formie potwierdzenia przez przodownika lub przewodnika, fotografii z datą, zapisu położenia z wykorzystaniem GPS. " typ = "ograniczenie operacji"	Text = "Odcinki opisane może tworzyć tylko Członek Komisji. " typ = "ograniczenie operacji"
<<BusinessRule>> Potwierdzanie wycieczki przez przodownika	<<BusinessRule>> Nadwyżka punktowa	<<BusinessRule>> Punkty opisane
Text = "Przodownik może potwierdzić turystę odbycie zaplanowanej wycieczki tylko jeśli brał on udział w tej wycieczce" typ = "ograniczenie operacji"	Text = "Nadwyżka punktów może być naliczona na poczet następnego stopnia małego tylko przy weryfikacji oznaki popularnej, malej brązowej i malej typ = "ograniczenie operacji"	Text = "Punkty opisane może tworzyć tylko Członek Komisji. " typ = "ograniczenie operacji"
<<BusinessRule>> Potwierdzanie wycieczki przez przewodnika	<<BusinessRule>> Odrzucone planowanie wycieczki	<<BusinessRule>> Pierwsze zdobywanie odznaki
Text = "Przewodnik może potwierdzić turystę odbycie zaplanowanej wycieczki tylko jeśli: ● ma uprawnienia na poszczególną grupę górską, nawet jeśli nie był na danej wycieczce ● poza zakresem swoich uprawnień jeśli osobiście brał udział w wycieczce" typ = "ograniczenie operacji"	Text = "Nie można planować wycieczki przez dany odcinek, jeśli jest on chwilowo nieczynny lub całkowicie zlikwidowany. " typ = "ograniczenie operacji"	Text = "Podczas pierwszego zdobywania określonej odznaki, punkty uzyskane na wycieczkach odbytych w grupach górskich Sudetów i Karpat położonych na obszarach krajów sąsiadujących z Polską nie mogą stanowić więcej niż 50 % punktów niezbędnych do zdobycia popularnego i małych stopni GOT PTTK. " typ = "ograniczenie operacji"
<<BusinessRule>> Zlikwidowany odcinek	<<BusinessRule>> Dodawanie odcinka własnego	<<BusinessRule>> Konfiguracja trasy
Text = "O uznaniu odcinka za zlikwidowany może decydować tylko Członek Komisji. " typ = "ograniczenie operacji"	Text = "Turysta musi wprowadzić różnicę wysokości punktu początkowego i końcowego oraz długość dodawanego typ = "ograniczenie operacji"	Text = "Punkt końcowy odcinka trasy musi być początkowym punktem kolejnego odcinka trasy lub końcem trasy w której skład wchodzi. " typ = "ograniczenie operacji"
<<BusinessRule>> Odcinek nieczynny	<<BusinessRule>> Weryfikacja statusu odcinka	
Text = "Odcinek może być zgłoszony jako nieczynny tylko przez przodownika lub przewodnika. " typ = "ograniczenie operacji"	Text = "Członek Komisji może zmienić status odcinka opisanego na nieczynny, jeśli zostanie zgłoszona taka potrzeba. " typ = "ograniczenie operacji"	

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

Visual Paradigm Standard (Justyna/Institute of Informatics))

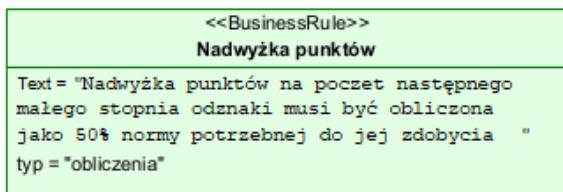
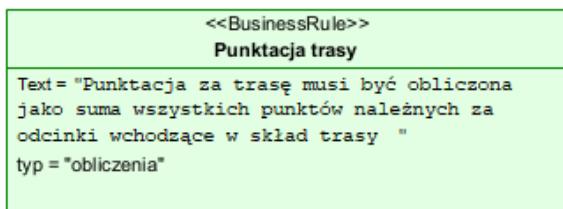
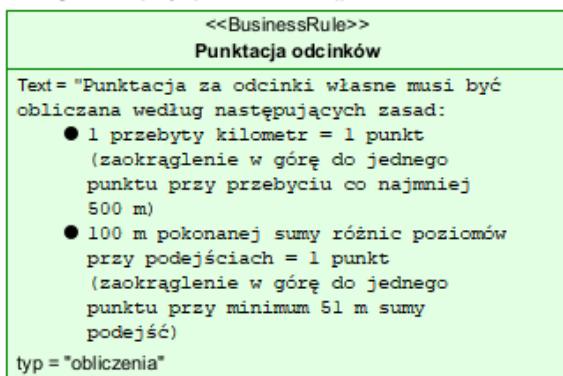


Visual Paradigm Standard (Justyna/Institute of Informatics))



“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

Visual Paradigm Standard(Justyna[Institute of Informatics])



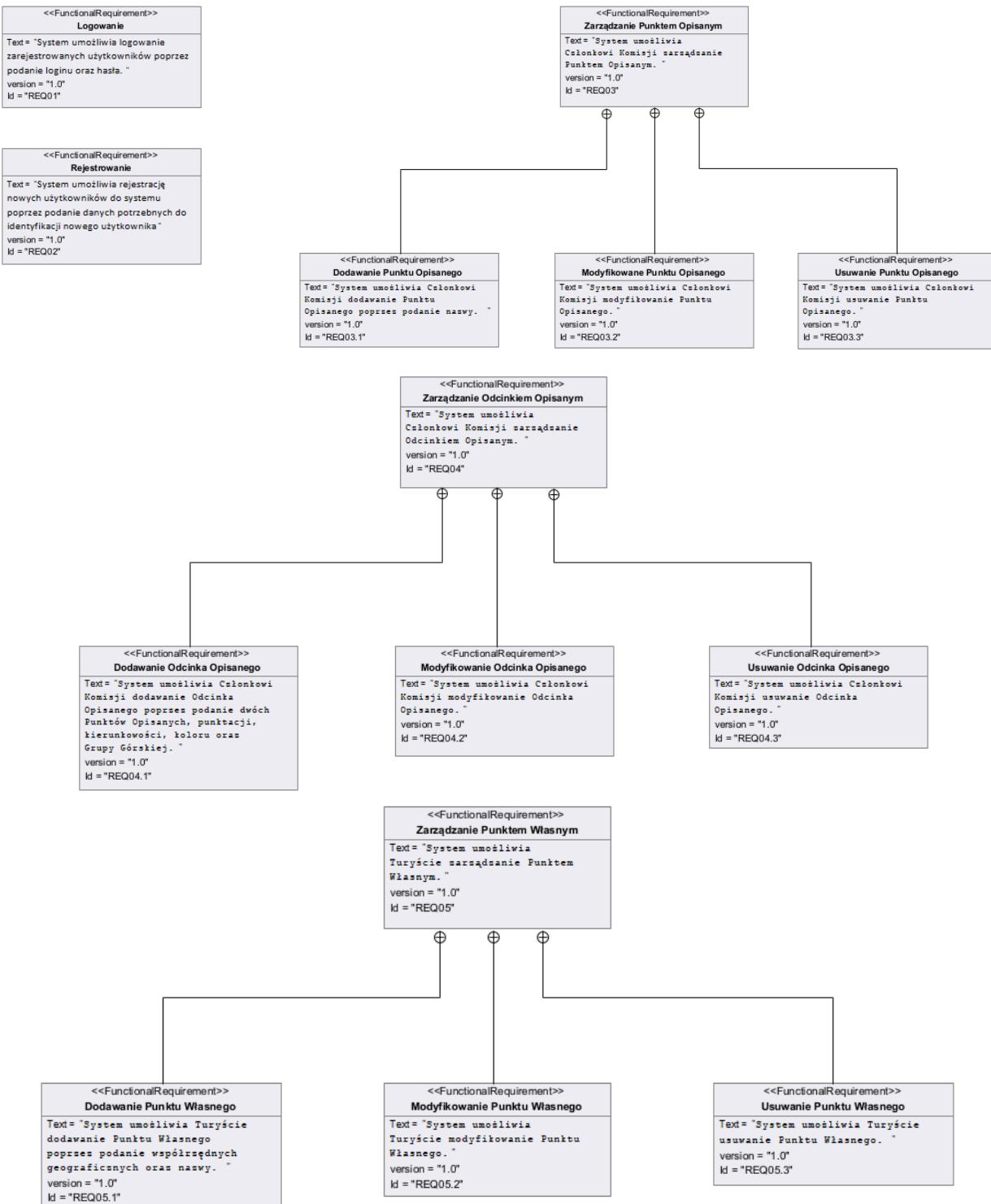
Etap II

Zmiany wprowadzone do etapu I:

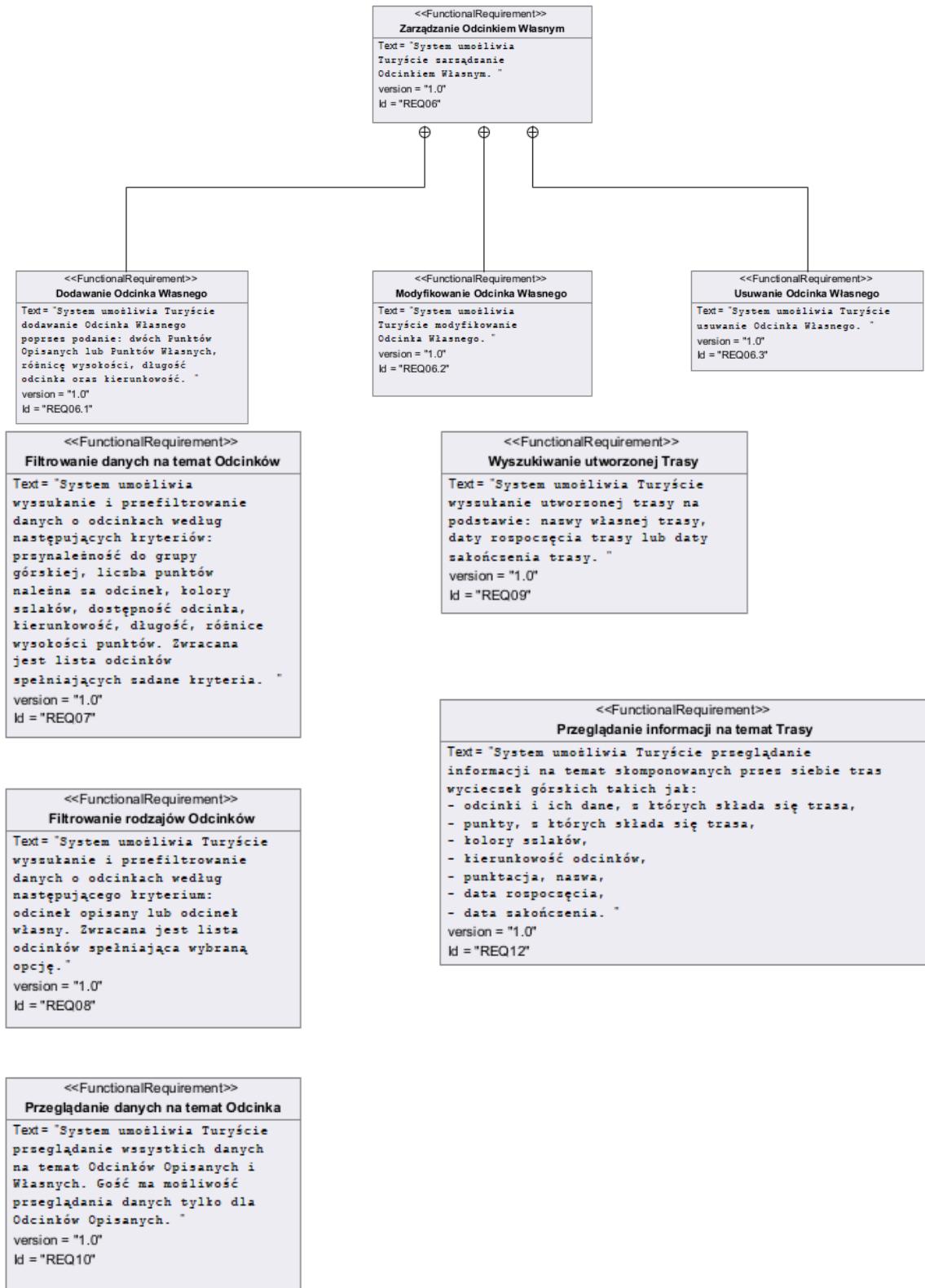
- Potrzeba 3 – zmiana „danych o trasach” na „danych o odcinkach”
- Potrzeba 3 - usunięcie słowa “grupowanie” z treści potrzeby i cechy
- Dodanie atrybutu Nazwa dla Trasy (słownik, oraz VP)
- Wyrzucamy potrzebę: “System przechowujący dane zaplanowanych tras oraz ich punktacji”
- Złączenie potrzeby “System wspierający proces weryfikacji poprzez GPS” z potrzebą “Załączanie dokumentacji do odznak (dowody przebycia trasy)”
- Dodanie atrybutu obowiązkowego “przez” do OdcinkaOpisanego + poprawienie opisu w Etapie 1 na ten temat
- Dodanie reguły biznesowej dotyczącej atrybutu “przez”
- Modyfikacja reguły dotyczącej nazwy punktu opisanego i punktu własnego
- Doprecyzowanie reguł biznesowych dotyczących numeru legitymacji Przewodnika i Przodownika

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

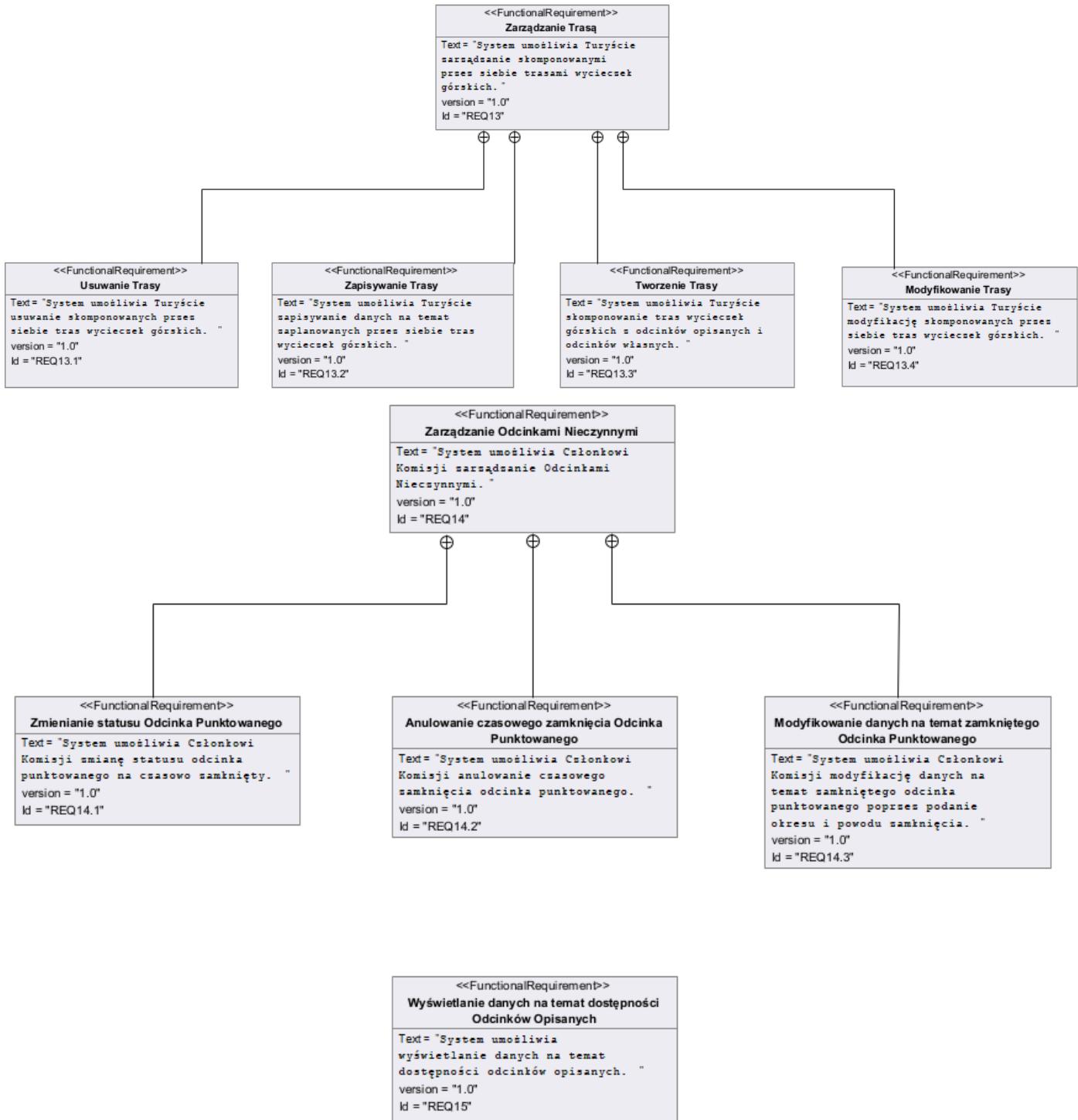
Specyfikacja wymagań



“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>



“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>



“Turysta Z Odznaką”<Project Name>		
Etap I		Data: <dd/mmm/yy>
<p><<FunctionalRequirement>> Eksportowanie danych na temat Trasy</p> <p>Text = "System umożliwia eksportowanie tras do formatu PDF." version = "1.0" Id = "REQ16"</p>	<p><<FunctionalRequirement>> Zmienianie statusu likwidowania Odcinka Punktowanego</p> <p>Text = "System umożliwia Członkowi Komisji zmianę statusu likwidowania Odcinka Punktowanego: likwidowany, czynny." version = "1.0" Id = "REQ17"</p>	<p><<FunctionalRequirement>> Automatyczne naliczanie punktów</p> <p>Text = "System umożliwia automatyczne naliczanie punktów Turystce za przebyte odcinki tras, po uprzednim zatwierdzeniu lub zweryfikowaniu dowodów przez Przewodnika lub Prasodownika oraz po spełnieniu warunków zawartych w regulaminie." version = "1.0" Id = "REQ19"</p>
<p><<FunctionalRequirement>> Wyświetlanie danych na temat Odcinków Zlikwidowanych</p> <p>Text = "System umożliwia wyświetlanie danych na temat odcinków zlikwidowanych." version = "1.0" Id = "REQ18"</p>	<p><<FunctionalRequirement>> Automatyczne naliczanie odznak</p> <p>Text = "System umożliwia automatyczne naliczanie odznak Turystce, po zdobyciu odpowiedniej liczby punktów i spełnieniu warunków zawartych w regulaminie." version = "1.0" Id = "REQ20"</p>	
<p><<FunctionalRequirement>> Wyszukiwanie Odznak</p> <p>Text = "System umożliwia Turystce wyszukanie odznaki na podstawie: rodzaju odznaki, stopnia odznaki, roku zdobycia. Zwrotnie jest lista posiadanych odznak, zgodnych z podanymi kryteriami." version = "1.0" Id = "REQ21"</p>	<p><<FunctionalRequirement>> Zgłoszenie prośby o zatwierdzenie przebycia Trasy</p> <p>Text = "System umożliwia Turystce zgłoszenie prośby o zatwierdzenie przebycia odcinków trasy do Przewodnika lub Prasodownika." version = "1.0" Id = "REQ23"</p>	
<p><<FunctionalRequirement>> Wyświetlanie danych na temat Odznaki</p> <p>Text = "System prezentuje wszystkie dane wybranej odznaki oraz dane wszystkich dowodów z nią powiązanych." version = "1.0" Id = "REQ22"</p>	<p><<FunctionalRequirement>> Przypisanie Prasodownika lub Przewodnika do prośby</p> <p>Text = "System umożliwia Turystce przypisanie Prasodownika lub Przewodnika odpowiedzialnego za rozpatrzenie prośby." version = "1.0" Id = "REQ24"</p>	
		<p><<FunctionalRequirement>> Zgłoszenie prośby o zweryfikowanie Dowodów</p> <p>Text = "System umożliwia Turystce zgłoszenie prośby o zweryfikowanie dowodów przebycia odcinków trasy do Prasodownika." version = "1.0" Id = "REQ25"</p>
		<p><<FunctionalRequirement>> Anulowanie prośby o zatwierdzenie Dowodów</p> <p>Text = "System umożliwia Turystce anulowanie wysłanej prośby o zatwierdzenie." version = "1.0" Id = "REQ26"</p>

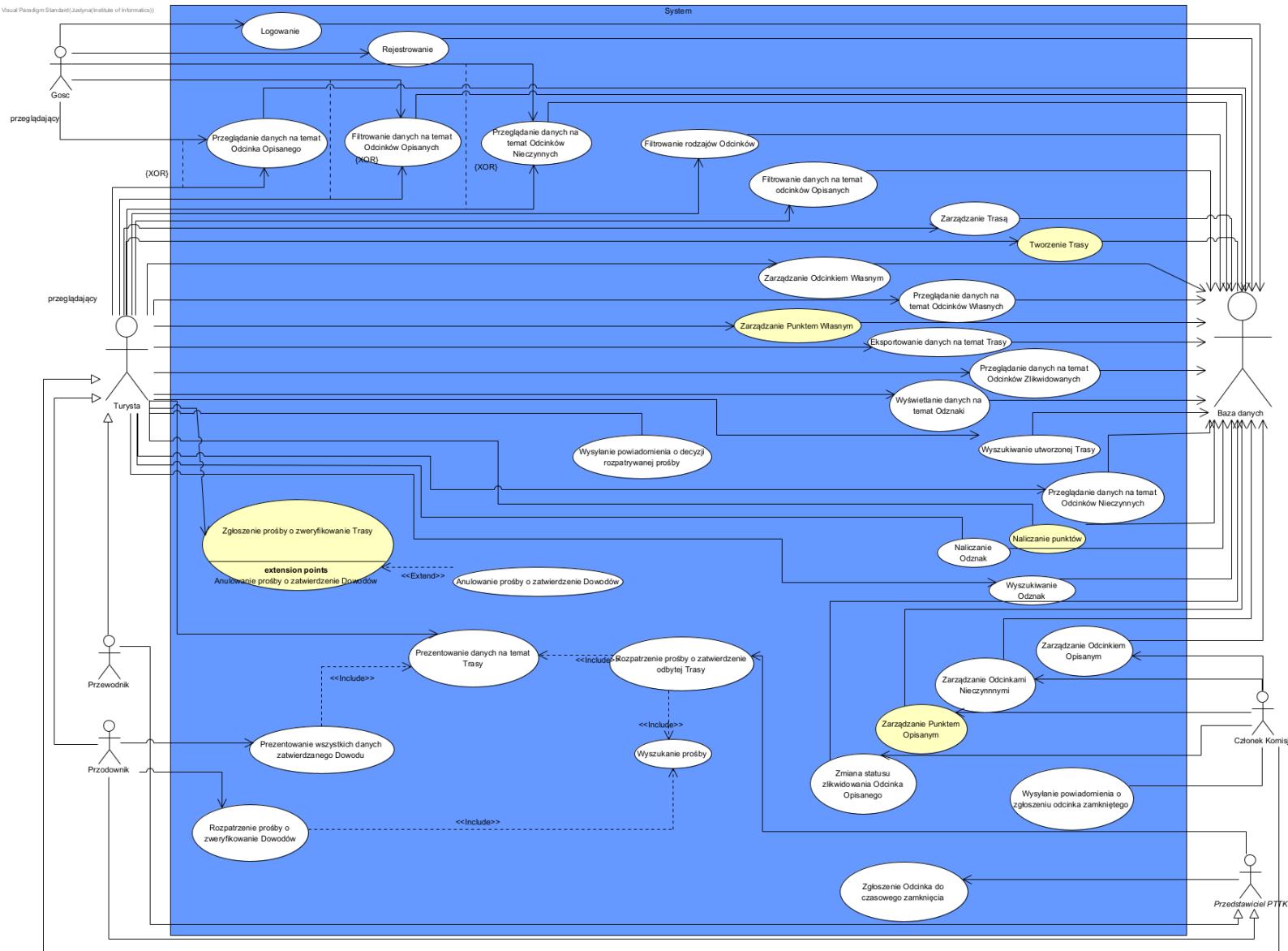
“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

<p><<FunctionalRequirement>> Rozpatrzenie prośby o zatwierdzenie odbytej Trasy</p> <pre>Text= "System umożliwia Przodownikowi lub Przewodnikowi rozpatrzenie prośby o zatwierdzenie odbytej trasy. " version = "1.0" Id = "REQ27"</pre>	<p><<FunctionalRequirement>> Zgłoszenie Odcinka Punktowanego, do czasowego zamknięcia</p> <pre>Text= "System umożliwia Przodownikowi i Przewodnikowi zgłoszenie odcinka punktowanego, do czasowego zamknięcia. " version = "1.0" Id = "REQ32"</pre>	<p><<FunctionalRequirement>> Złączenie dowodu przebycia Odcinków</p> <pre>Text= "System umożliwia turystie załączenie dowodu przebycia odcinka lub odcinków w odpowiedniej formie, tj. fotografia lub mapę polożenia GPS." version = "1.0" Id = "REQ34"</pre>
<p><<FunctionalRequirement>> Wyszukanie prośby o zatwierdzenie odbytej Trasy</p> <pre>Text= "System umożliwia Przodownikowi i Przewodnikowi wyszukanie prośby o zatwierdzenie na podstawie: identyfikatora turysty lub daty wysłania prośby. Zwrotnica jest listą aktualnych i nierostrzygniętych prośb. " version = "1.0" Id = "REQ28"</pre>	<p><<FunctionalRequirement>> Wysyłanie powiadomienia o nowym zgłoszeniu odcinka zamkniętego</p> <pre>Text= "System wysyła do Członka Komisji powiadomienie informujące o nowym zgłoszeniu odcinka zamkniętego. " version = "1.0" Id = "REQ33"</pre>	
<p><<FunctionalRequirement>> Rozpatrzenie prośby o zweryfikowanie Dowodów</p> <pre>Text= "System umożliwia Przodownikowi zweryfikowanie prośby o zatwierdzenie Dowodów przebytej trasy. " version = "1.0" Id = "REQ29"</pre>		
	<p><<FunctionalRequirement>> Prezentowanie wszystkich danych na temat przebytej Trasy, której dotyczy prośba</p> <pre>Text= "System prezentuje Przodownikowi wszystkie dane przebytej trasy, której dotyczy prośba oraz przypisane do niej dowody, poswalające na wydanie decyzji. " version = "1.0" Id = "REQ30"</pre>	
	<p><<FunctionalRequirement>> Wydawanie decyzji do prośby o zatwierdzenie</p> <pre>Text= "System umożliwia Przodownikowi i Przewodnikowi wydanie decyzji do prośby o zatwierdzenie: akceptację, odrzucenie. " version = "1.0" Id = "REQ30"</pre>	
	<p><<FunctionalRequirement>> Wysyłanie powiadomienie informującego o decyzji w sprawie prośby</p> <pre>Text= "System wysyła do Turysty powiadomienie informujące o zaakceptowaniu lub odrzuceniu wniosku. " version = "1.0" Id = "REQ31"</pre>	

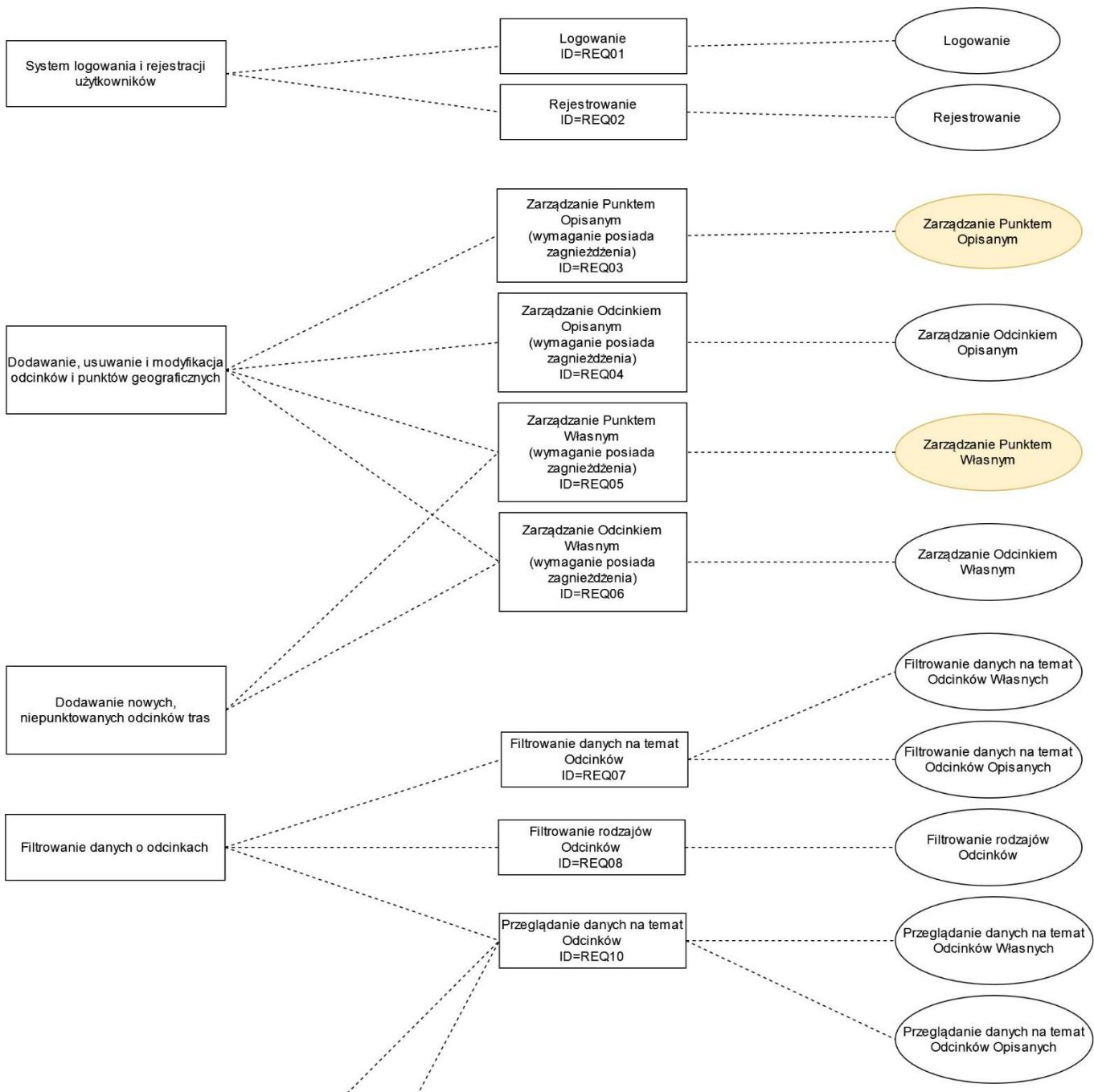
“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

Model przypadków użycia i prototypy interfejsów

7. Diagram przypadków użycia



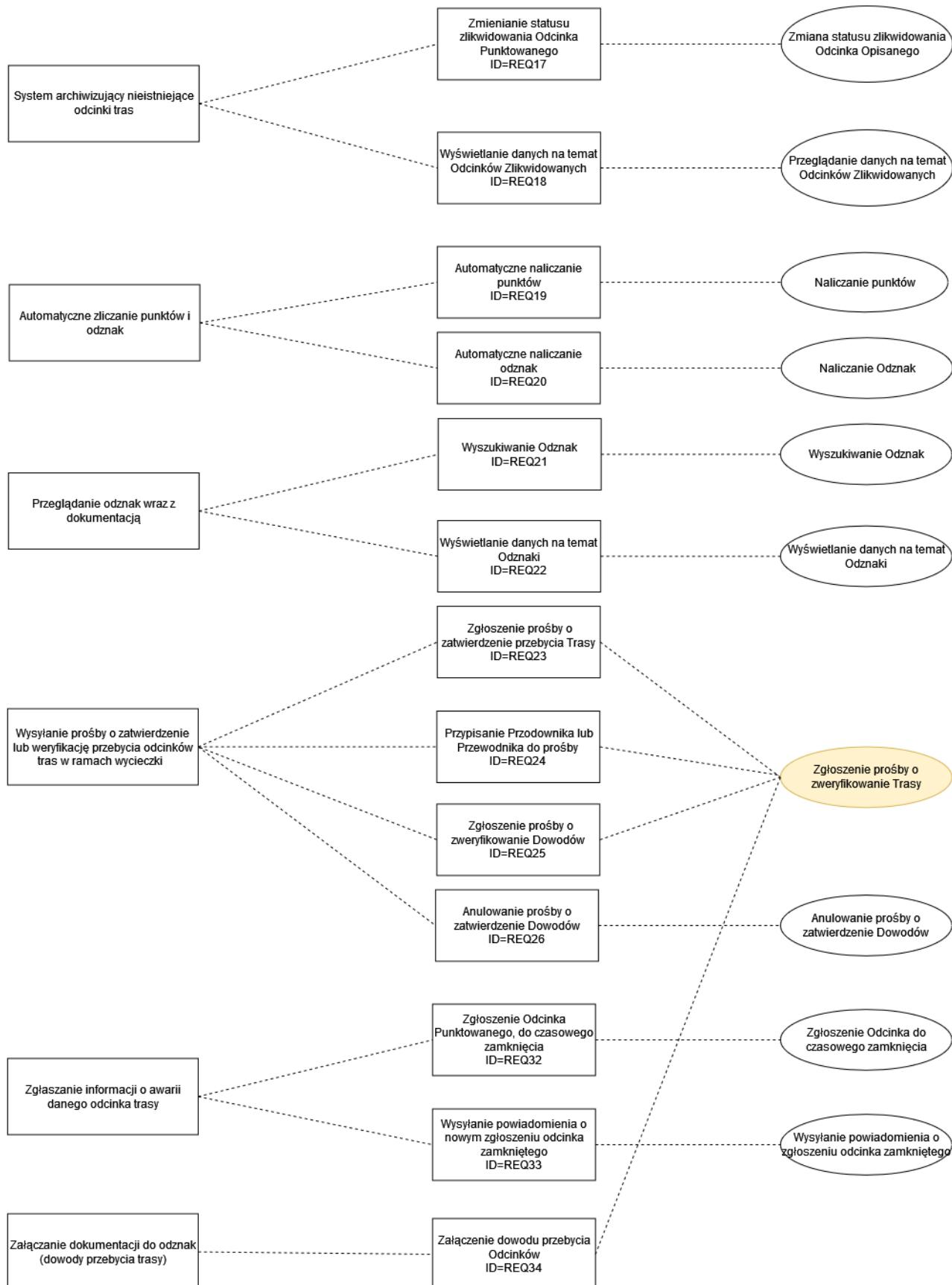
“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>



“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>



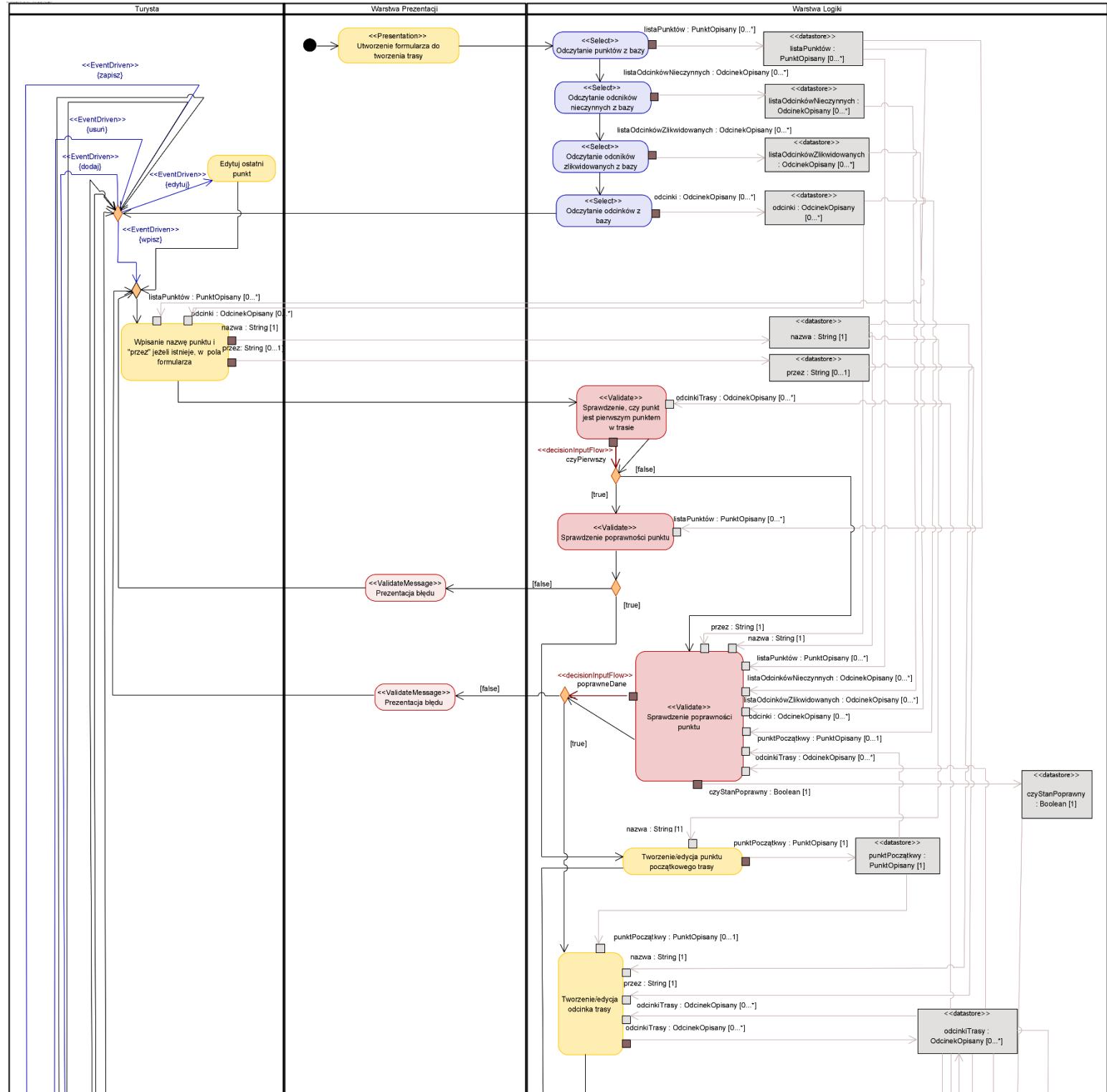
“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>



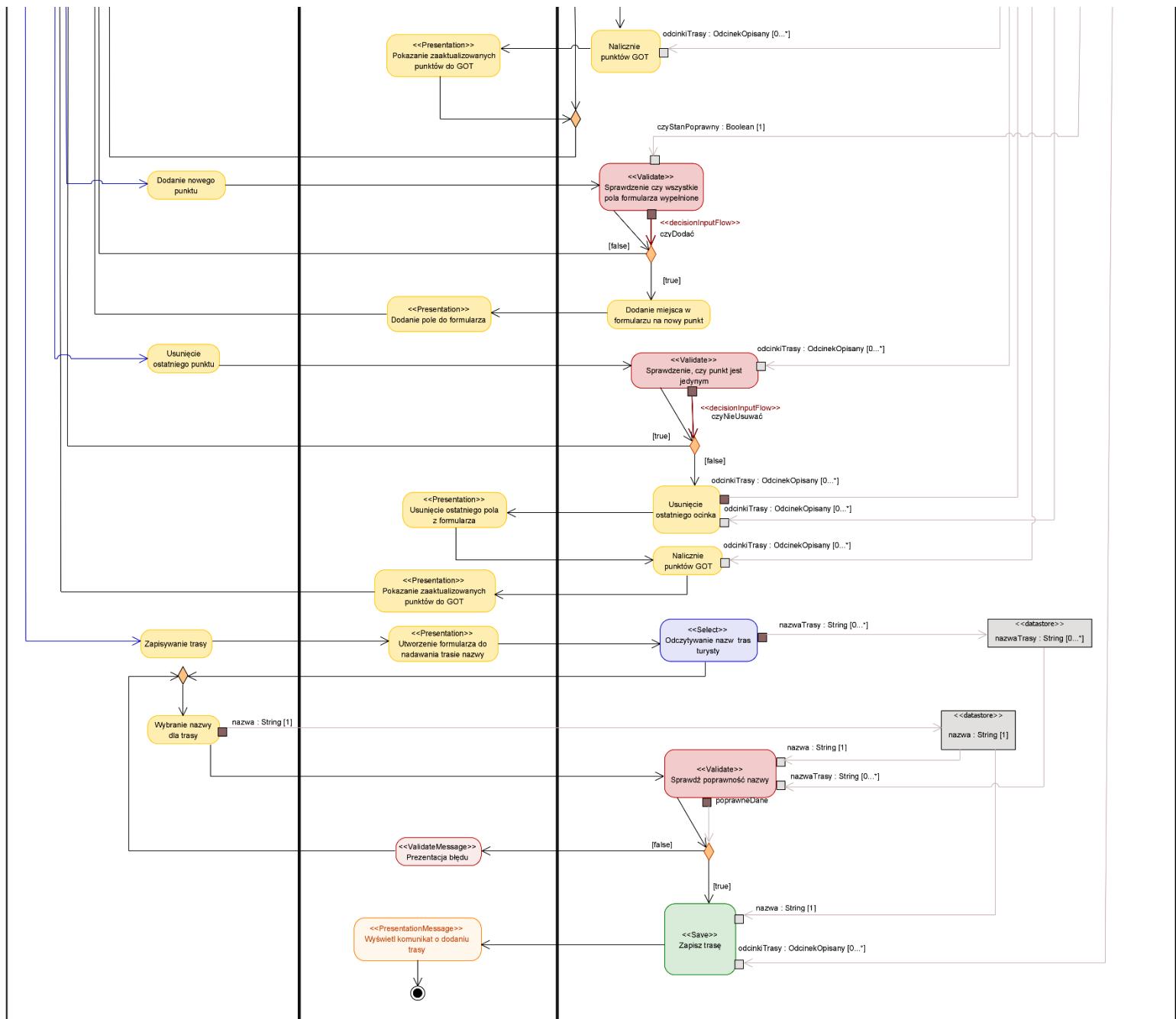
8. Specyfikacja przypadków użycia

8.1.1 Tworzenie Trasy (Klaudia Błażyczek)

Skomponowanie trasy wycieczki górskiej z Odcinków Własnych i Odcinków Opisanych.



“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>



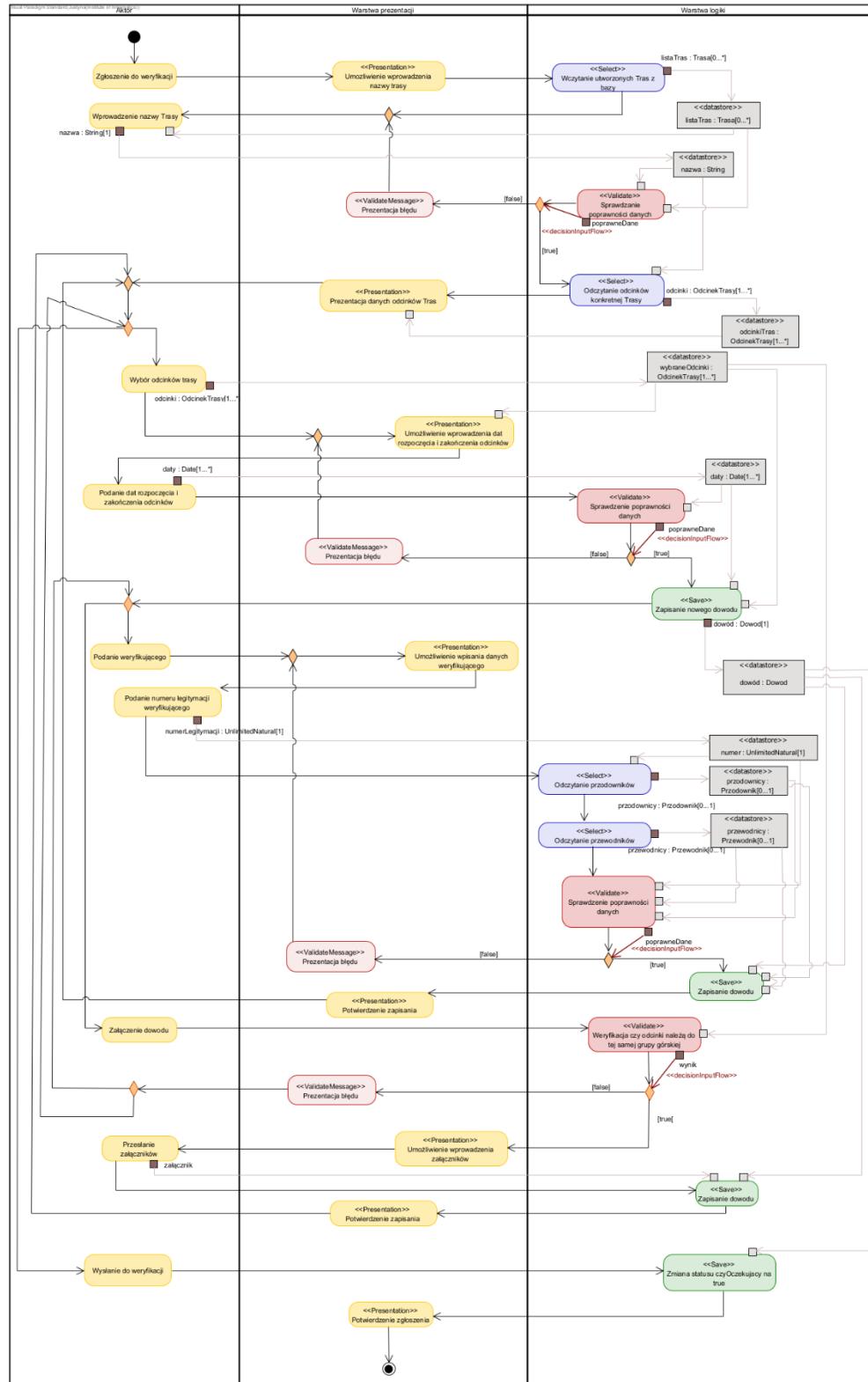
(Niektóre akcje mają dodany komentarz w opisie specyfikacji w VP)

Uproszczenie wprowadzone w porozumieniu z prowadzącym: trasę można stworzyć tylko z punktów opisanych

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

8.1.2 Zgłaszcenie prośby o zweryfikowanie Trasy (Justyna Małuszyńska)

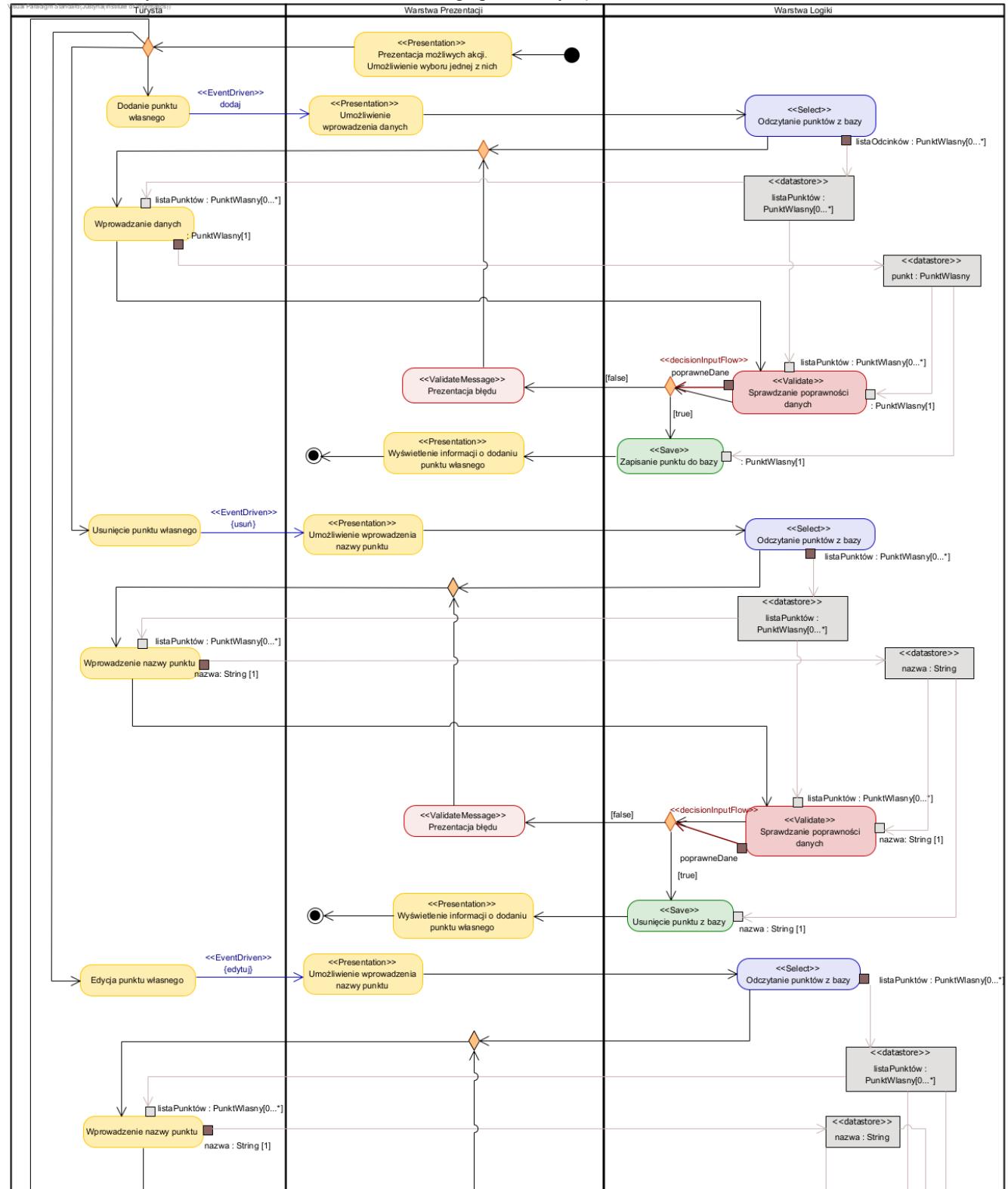
Zgłoszenie wybranych odcinków z Trasy do zatwierdzenia poprzez potwierdzenie Przewodnika/Przodownika lub dowód przejścia Trasy



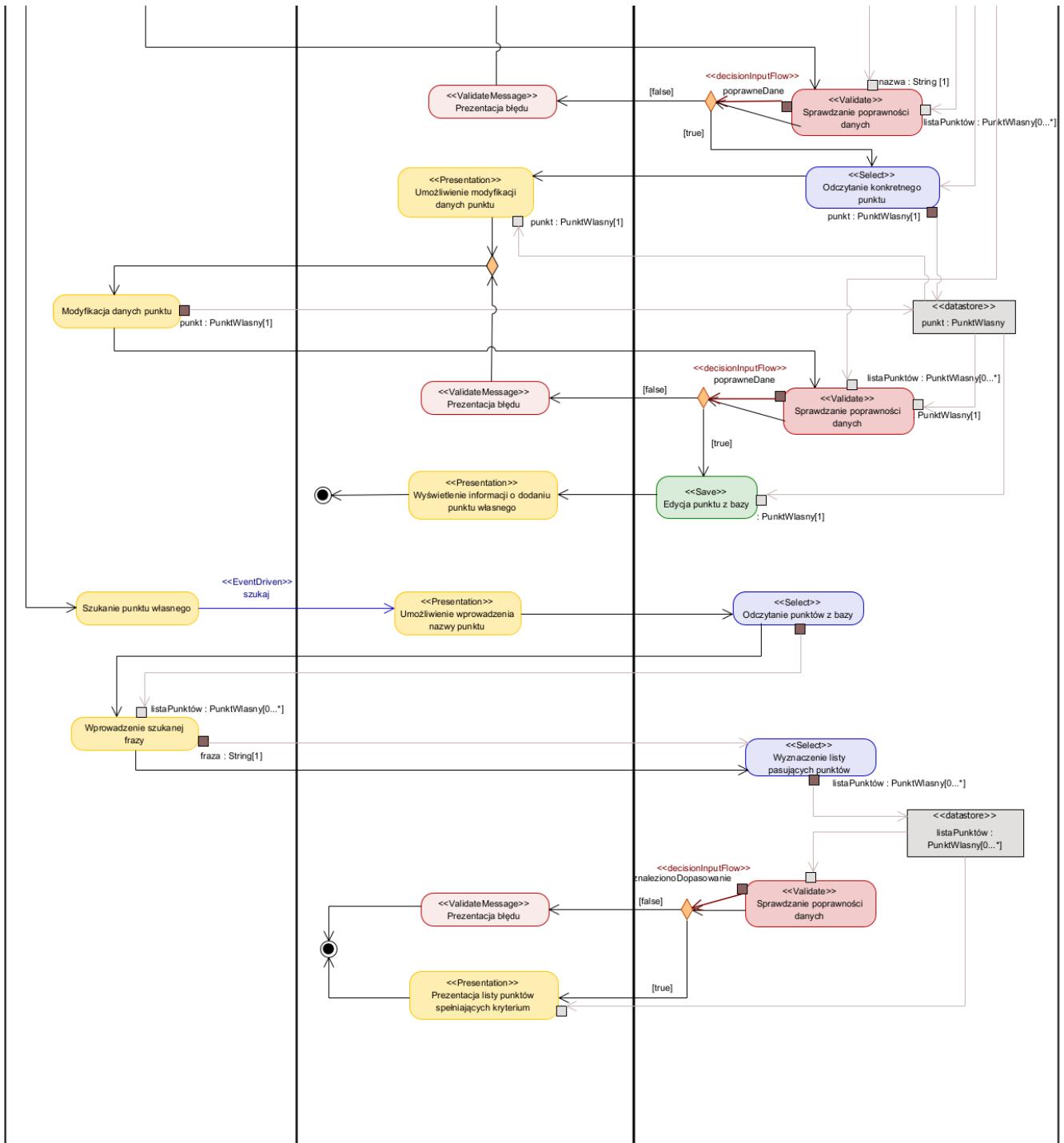
“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

8.1.3 Zarządzanie Punktami Własnymi (Justyna Małuszyńska)

Dodawanie, modyfikowanie i usuwanie Punktu Własnego przez Turystę.



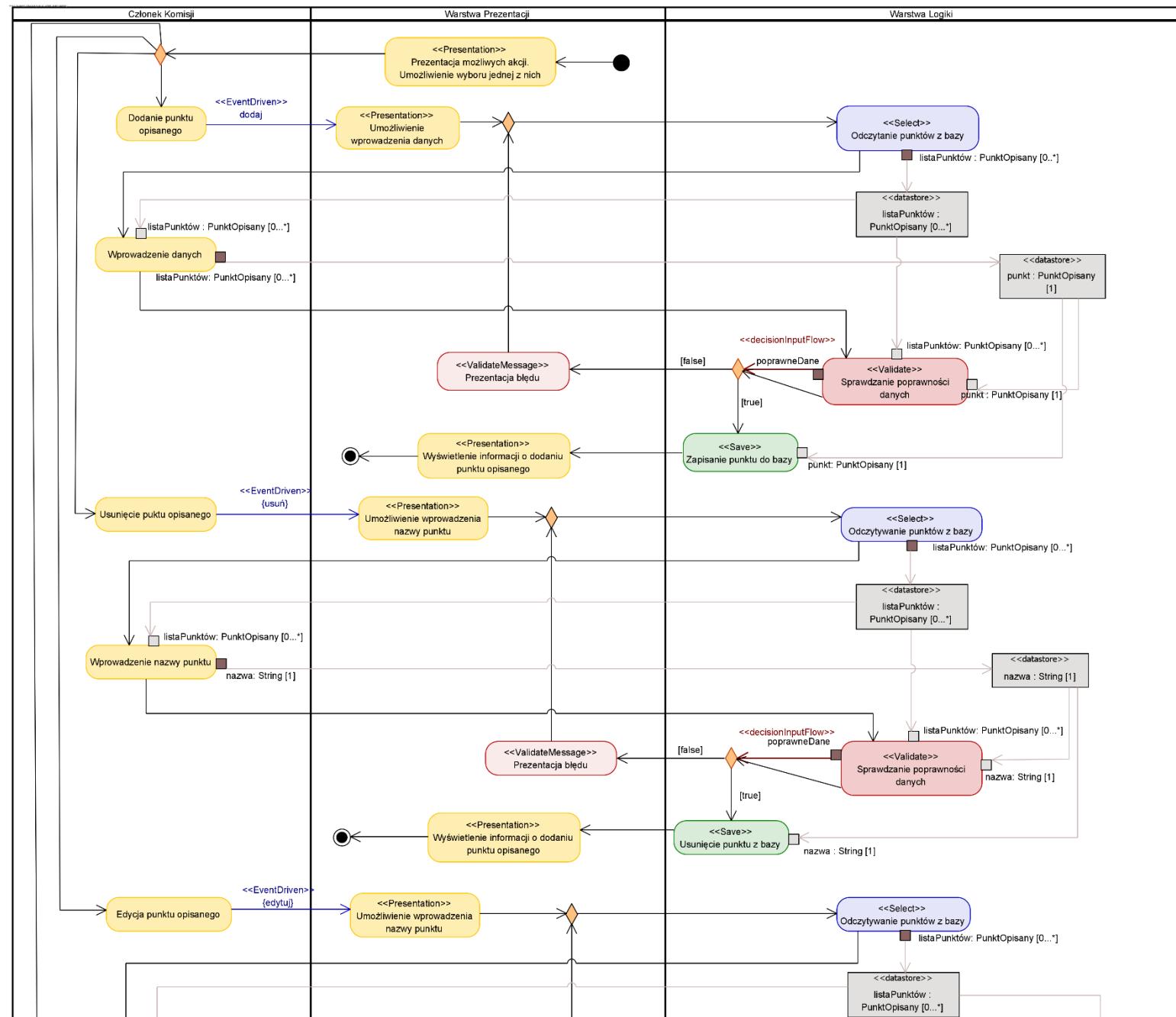
“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

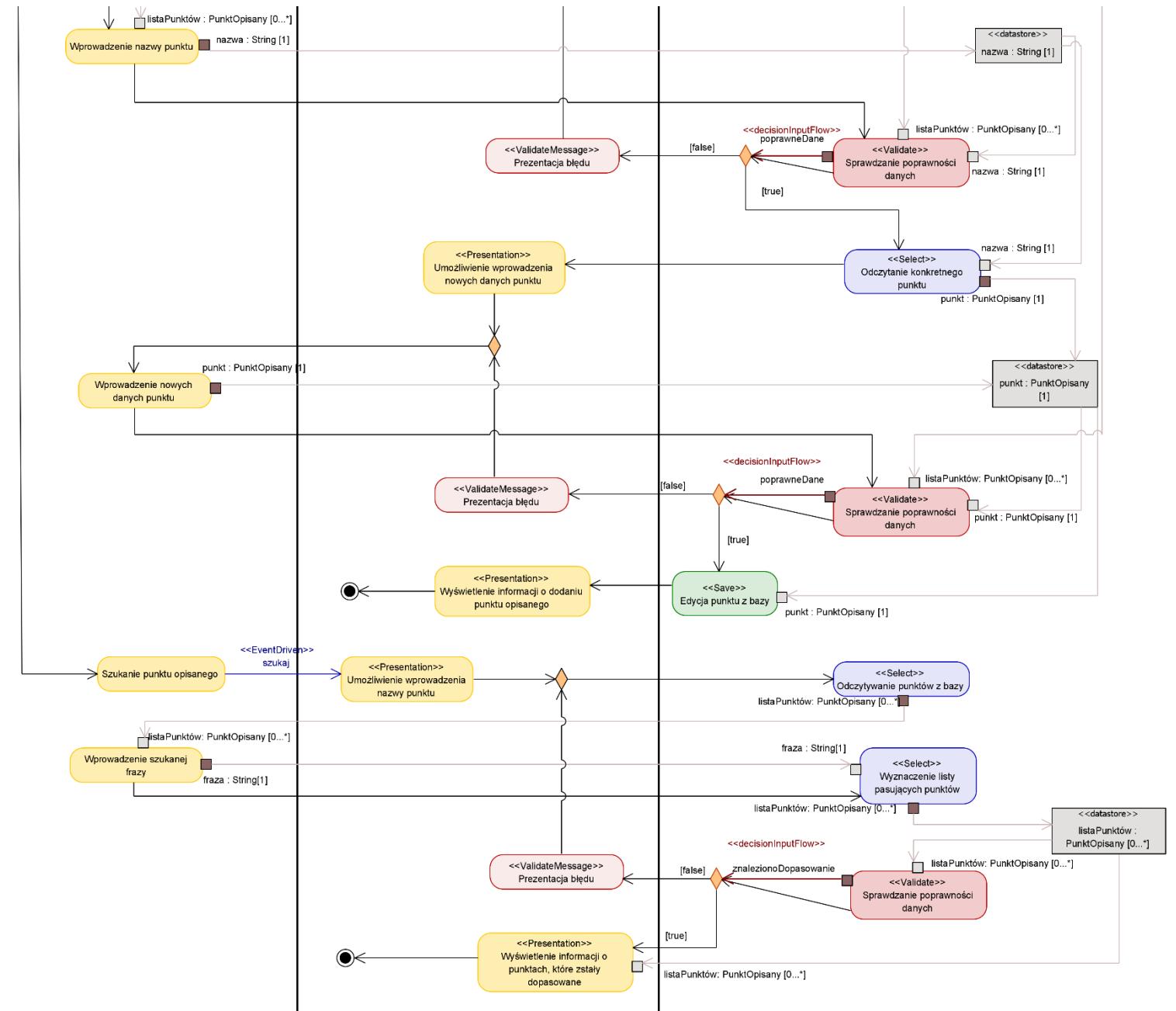


Uproszczenie/ustalenie wprowadzone w porozumieniu z prowadzącym: punkt własny można edytować i usuwać tylko, gdy nie został jeszcze użyty do stworzenia żadnego odcinka.

8.1.4 Zarządzanie Punktem Opisanym (Klaudia Błażyczek)

Dodawanie, modyfikowanie i usuwanie Punktu Opisanego przez Członka Komisji.





Uproszczenie/ustalenie wprowadzone w porozumieniu z prowadzącym: punkt własny można edytować i usuwać tylko, gdy nie został jeszcze użyty do stworzenia żadnego odcinka.

9. Prototyp interfejsu

9.1.1 Reguły przewodnika stylu

- Kolorystyka nawiązująca do tematyki górskiej oraz natury.
- Preferowane kolory: odcienie zieleni
- Preferowane odcienie zieleni:
 - Jasny: #82B479

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

- Ciemny: #82B479 z nałożonym drop shadow
- Preferowany odcień pomarańczowy: #DD9549
- Kolory przycisków:
 - W stanie spoczynkowym: białe tło, pomarańczowy tekst i obwódka
 - W stanie hover: pomarańczowe tło i obwódka, biały tekst
- Dostęp do menu nawigacyjnego jest na każdej podstronie.
- Menu nawigacyjne wyświetcone po lewej stronie, w odcieniach zieleni.
- Pasek z wyszukiwaniem dostępny jest na każdej stronie, u góry
- Czcionka obowiązująca w menu nawigacyjnym: Nirmala UI. Czcionka obowiązująca dla całej witryny: Source Sans Pro.
- Dopuszcza się zmianę czcionek na stronie. Muszą być one jednak bezszeryfowe oraz być uznawane za czytelne, łatwe w czytaniu. Czcionki te powinny też udostępniać przynajmniej dwa rodzaje grubości: Regular oraz Bold
- Komunikaty o błędach powinny być koloru czerwonego
- W przypadku komunikatów o błędach, zalecany kolor czerwieni to: #FF6565
- Obrazy umieszczane w tle strony powinny kolorystycznie pasować do kolorystyki samej strony

9.1.2 Tworzenie Trasy

Tworzenie trasy - wątek główny

The screenshot shows the 'Tworzenie trasy' (Create Route) page. The left sidebar is titled 'TURYSTA Z ODZNAKĄ' and contains the following items:

- Strona główna
- Powiadomienia
- Zarządzanie Punktem Opisanym
- Zarządzanie Punktem Własnym
- Tworzenie trasy** (highlighted)
- Settings

The main content area has a search bar labeled 'Wyszukaj' and a user profile for 'Jan Kowalski'. Below that, it says 'Stwórz trasę poprzez podanie nazw punktów wchodzących w skład trasy'. There are two entries:

- Nazwa Punktu: Wetlina (with a green flag icon)
- Nazwa Punktu: Przełęcz Ort (with an orange flag icon)

A button labeled 'Zapisz trasę' is located at the bottom right of the input area. To the right, there is a box labeled 'Punkty do GOT' with a value of '0'.

Tworzenie trasy - wątek główny - start

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

TURYSTA Z ODZNAKĄ

Strona główna

Powiadomienia

Zarządzanie Punktami Opisanym

Zarządzanie Punktami Własnym

Tworzenie trasy

Settings

Wyszukaj

Nazwa Punktu

Wetlina

Nazwa Punktu

Przełęcz Orłowicza

+ Dodaj kolejny punkt

Zapisz trasę

Punkty do GOT
9

Tworzenie trasy - wątek główny - start – uzupełnienie

TURYSTA Z ODZNAKĄ

Strona główna

Powiadomienia

Zarządzanie Punktami Opisanym

Zarządzanie Punktami Własnym

Tworzenie trasy

Settings

Wyszukaj

Nazwa Punktu

Wetlina

Nazwa Punktu

Przełęcz Orłowicza

Nazwa Punktu

Smerek

Kryszowa

Polonina Wetlińska

Zapisz trasę

Punkty do GOT
9

Tworzenie trasy - wątek główny – dodanie kolejnego punktu - przykład

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

Strona główna

Powiadomienia

Zarządzanie Punktem Opisanym

Zarządzanie Punktem Własnym

Tworzenie trasy

Settings

Wyszukaj

Stwórz trasę poprzez podanie nazw punktów wchodzących w skład trasy

Nazwa Punktu

Wetlina

Nazwa Punktu

Przełęcz Orłowicza

Nazwa Punktu

Połonina Wetlińska

Dodaj kolejny punkt

Zapisz trasę

Punkty do GOT
14

Tworzenie trasy - wątek główny – dodanie kolejnego punktu - przykład – 2

Strona główna

Powiadomienia

Zarządzanie Punktem Opisanym

Zarządzanie Punktem Własnym

Tworzenie trasy

Settings

Wyszukaj

Stwórz trasę poprzez podanie nazw punktów wchodzących w skład trasy

Nazwa Punktu

Wetlina

Nazwa Punktu

Przełęcz Orłowicza

Nazwa Punktu

Połonina Wetlińska

Dodaj kolejny punkt

Zapisz trasę

Punkty do GOT
9

Tworzenie trasy - wątek główny – usuniecie jakiegoś punktu z trasy - przykład

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

Tworzenie trasy - wątek główny – nadanie nazwy trasie

Tworzenie trasy - wątek główny – koniec

Usuwać i edytować można tylko ostatni punkt – do celów utrzymania ciągłości trasy

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

Wątki alternatywne – plansze dla alternatywnych wątków

Tworzenie trasy - wątek alternatywny - podanie istniejącej nazwy trasy – nadanie istniejącej nazwy

W przypadku powyższego wątku alternatywnego naduszenie przycisku “Zatwierdź” będzie nimożliwe, dopóki podawana będzie nazwa trasy, która już istnieje.

Po poprawnym wypełnieniu formularza, po naciśnięciu przycisku, przeniesieni zostaniemy do makietы: “Z Tworzenie trasy - wątek główny – nadanie nazwy trasie ”

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

Tworzenie trasy - wątek alternatywny podanie nieistniejącego punktu - start – zły punkt

W przypadku powyższego wątku alternatywnego naduszenie przycisku “Zapisz trasę”, dodanie i usuwanie punktów będzie nimożliwe, dopóki podawany punkt nie istnieje.

Po poprawnym wypełnieniu formularza, można wykonywać dalsze operacje. (zapisanie trasy, dodanie/edykcja/usunięcie punktów)

Tworzenie trasy - wątek alternatywny podanie punktów nie tworzących odcinki - start

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

W przypadku powyższego wątku alternatywnego naduszenie przycisku “Zapisz trasę”, dodanie, usuwanie punktów będzie nimożliwe, dopóki podawany punkt nie tworzy odcinka z poprzednim.

Po poprawnym wypełnieniu formularza, można wykonywać dalsze operacje. (zapisanie trasy, dodanie/edykcja/usunięcie punktów)

Tworzenie trasy - wątek alternatywny i wyjątkowy - podanie odcinka nieczynnego/zlikwidowanego - start

W przypadku powyższego wątku alternatywnego naduszenie przycisku “Zapisz trasę”, dodanie i usuwanie punktów będzie nimożliwe, dopóki podawany punkt nie tworzy odcinka z poprzednim.

Po poprawnym wypełnieniu formularza, można wykonywać dalsze operacje. (zapisanie trasy, dodanie/edykcja/usunięcie punktów)

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

9.1.3 Zgłaszanie prośby o zweryfikowanie Trasy

The screenshot shows the application's main menu on the left with the title "TURYSTA Z ODZNAKĄ". The menu items include: Strona główna, Powiadomienia, Zarządzanie Punktem Opisanym, Zarządzanie Punktem Własnym, Tworzenie trasy, Settings, and Zgłoszenie do weryfikacji. On the right, there is a search bar with the placeholder "Wyszukaj", a user profile for "Jan Kowalski", and a large circular icon featuring a hiker walking with a backpack and trekking poles against a mountainous background.

Zgłaszanie prośby o zweryfikowanie Trasy – ekran startowy

This screenshot shows the same application interface as above, but with the "Zgłoszenie do weryfikacji" menu item selected. In the search bar, the text "Trasa Bieszczady Lato 2021" is typed. The rest of the interface, including the user profile and the circular hiker icon, remains the same.

Zgłaszanie prośby o zweryfikowanie Trasy – wpisanie nazwy trasy (naduszenie przycisku możliwe po wprowadzeniu poprawnej nazwy trasy).

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

TURYSTA Z ODZNAKĄ

Wyszukaj

Wybierz odcinki do weryfikacji

Trasa Bieszczady Lato 2021

Lp.	POCZĄTEK	KONIEC	GRUPA GÓRSKA	LICZBA PUNKTÓW
1.	Wolosate	Przełęcz Bukowska	Bieszczady	8
2.	Przełęcz Bukowska	Rozsypaniec	Bieszczady	6
3.	Rozsypaniec	Halicz	Bieszczady	12
4.	Halicz	Przełęcz Goprowska	Bieszczady	8

Dalej

Dodane załączniki: 0
Przypisani weryfikujący: 0

Zgłoś

Zgłaszanie prośby o zweryfikowanie Trasy – wyświetlenie odcinków Trasy (naduszenie przycisku “Zgłoś” możliwe tylko gdy dodany zostanie załącznik lub przypisany weryfikujący).

TURYSTA Z ODZNAKĄ

Wyszukaj

Wybierz odcinki do weryfikacji

Trasa Bieszczady Lato 2021

Lp.	POCZĄTEK	KONIEC	GRUPA GÓRSKA	LICZBA PUNKTÓW
1.	Wolosate	Przełęcz Bukowska	Bieszczady	8
2.	Przełęcz Bukowska	Rozsypaniec	Bieszczady	6
3.	Rozsypaniec	Halicz	Bieszczady	12
4.	Halicz	Przełęcz Goprowska	Bieszczady	8

Dalej

Dodane załączniki: 0
Przypisani weryfikujący: 0

Zgłoś

Zgłaszanie prośby o zweryfikowanie Trasy – wybór odcinków Trasy do weryfikacji

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

TURYSTA Z ODZNAKĄ

Wyszukaj

Zadeklaruj daty rozpoczęcia i zakończenia odcinka

1. Wołosate -> Przełęcz Bukowska

Data rozpoczęcia 03.07.2021	Data zakończenia 03.07.2021
--------------------------------	--------------------------------

2. Przełęcz Bukowska -> Rozsypaniec

Data rozpoczęcia 04.07.2021	Data zakończenia 04.07.2021
--------------------------------	--------------------------------

Zapisz daty

Zgłaszanie prośby o zweryfikowanie Trasy – wprowadzenie dat

TURYSTA Z ODZNAKĄ

Wyszukaj

Wybierz odcinki do weryfikacji

Trasa Bieszczady Lato 2021

L.P.	POCZĄTEK	KONIEC	GRUPA GÓRSKA	LICZBA PUNKTÓW
1.	Wołosate	Przełęcz Bukowska	Bieszczady	8
2.	Przełęcz Bukowska	Rozsypaniec	Bieszczady	6
3.	Rozsypaniec	Halicz	Bieszczady	12
4.	Halicz	Przełęcz Goprowska	Bieszczady	8

Dodaj załącznik

Przypisz weryfikującego

Dodane załączniki: 0
Przypisani weryfikujący: 0

Zgłoś

Zgłaszanie prośby o zweryfikowanie Trasy – wybrane odcinki – wybór rodzaju dowodu przebycia odcinków

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

TURYSTA Z ODZNAKĄ

- Strona główna
- Powiadomienia
- Zarządzanie Punktem Opisanym
- Zarządzanie Punktem Własnym
- Tworzenie trasy
- Settings
- Zgłoszenie do weryfikacji

Wyszukaj

Dodaj załącznik

Zdjęcie, wypis GPS etc.

Wybierz

Dodaj

Zgłaszanie prośby o zweryfikowanie Trasy – dodawanie załącznika

TURYSTA Z ODZNAKĄ

- Strona główna
- Powiadomienia
- Zarządzanie Punktem Opisanym
- Zarządzanie Punktem Własnym
- Tworzenie trasy
- Settings
- Zgłoszenie do weryfikacji

Wyszukaj

Wybierz odcinki do weryfikacji

Trasa Bieszczady Lato 2021

L.P.	POCZĄTEK	KONIEC	GRUPA GÓRSKA	LICZBA PUNKTÓW
1.	Wolosate	Przełęcz Bukowska	Bieszczady	8
2.	Przełęcz Bukowska	Rozsypaniec	Bieszczady	6
3.	Rozsypaniec	Halicz	Bieszczady	12
4.	Halicz	Przełęcz Goprowska	Bieszczady	8

Dalej

Dodane załączniki: 1
Przypisani weryfikujący: 0

Zgłoś

Zgłaszanie prośby o zweryfikowanie Trasy – potwierdzenie pomyślnego dodania załącznika do wybranych odcinków, powrót do wyboru kolejnych odcinków lub zgłoszenia prośby

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

TURYSTA Z ODZNAKĄ

Wyszukaj

Wybierz odcinki do weryfikacji

Trasa Bieszczady Lato 2021

Lp.	POCZĄTEK	KONIEC	GRUPA GÓRSKA	LICZBA PUNKTÓW
1.	Wolosate	Przełęcz Bukowska	Bieszczady	8
2.	Przełęcz Bukowska	Rozsypaniec	Bieszczady	6
3.	Rozsypaniec	Halicz	Bieszczady	12
4.	Halicz	Przełęcz Goprowska	Bieszczady	8

Dalej

Dodane załączniki: 1
Przypisani weryfikujący: 0

Zgłoś

Zgłaszanie prośby o zweryfikowanie Trasy – wybór kolejnych odcinków

TURYSTA Z ODZNAKĄ

Wyszukaj

Zadeklaruj daty rozpoczęcia i zakończenia odcinka

3. Halicz -> Przełęcz Goprowska

Data rozpoczęcia
04.07.2021

Data zakończenia
04.07.2021

Zapisz daty

Zgłaszanie prośby o zweryfikowanie Trasy – zadeklarowanie dat

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

TURYSTA Z ODZNAKĄ

Wyszukaj

Wybierz odcinki do weryfikacji

Trasa Bieszczady Lato 2021

Lp.	POCZĄTEK	KONIEC	GRUPA GÓRSKA	LICZBA PUNKTÓW
1.	Wolosate	Przełęcz Bukowska	Bieszczady	8
2.	Przełęcz Bukowska	Rozsypaniec	Bieszczady	6
3.	Rozsypaniec	Halicz	Bieszczady	12
4.	Halicz	Przełęcz Goprowska	Bieszczady	8

Dodaj załącznik

Przypisz weryfikującego

Dodane załączniki: 1
Przypisani weryfikujący: 0

Zgłoś

Zgłaszanie prośby o zweryfikowanie Trasy – wybór rodzaju dowodu potwierdzającego przebycie wybranego odcinka

TURYSTA Z ODZNAKĄ

Wyszukaj

Wpisz dane przewodnika odbywającego z Tobą Trasę

Numer legitymacji
12345

Wybierz

Zgłaszanie prośby o zweryfikowanie Trasy – wprowadzenie numeru legitymacji

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

TURYSTA Z ODZNAKĄ

Wyszukaj

Wybierz odcinki do weryfikacji

Trasa Bieszczady Lato 2021

Lp.	POCZĄTEK	KONIEC	GRUPA GÓRSKA	LICZBA PUNKTÓW
1.	Wolosate	Przełęcz Bukowska	Bieszczady	8
2.	Przełęcz Bukowska	Rozsypaniec	Bieszczady	6
3.	Rozsypaniec	Halicz	Bieszczady	12
4.	Halicz	Przełęcz Goprowska	Bieszczady	8

Dodaj załącznik

Przypisz weryfikującego

Dodane załączniki: 1
Przypisani weryfikujący: 1

Zgłoś

Zgłaszanie prośby o zweryfikowanie Trasy – zgłaszczenie prośby

TURYSTA Z ODZNAKĄ

Wyszukaj

Zgłoszenie przebiegło pomyślnie, oczekuj na weryfikację!

Zakończ

Zgłaszanie prośby o zweryfikowanie Trasy – potwierdzenie zgłoszenia prośby

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

TURYSTA Z ODZNAKĄ

Wyszukaj

Strona główna

Powiadomienia

Zarządzanie Punktem Opisanym

Zarządzanie Punktem Własnym

Tworzenie trasy

Settings

Zgłoszenie do weryfikacji

Wybierz trasę do weryfikacji

Nazwa Szukanej Trasy
Trasa Bieczady Lato

Nie istnieje Trasa o wybranej nazwie

Szukaj



Zgłaszanie prośby o zweryfikowanie Trasy – błędna nazwa trasy

TURYSTA Z ODZNAKĄ

Wyszukaj

Strona główna

Powiadomienia

Zarządzanie Punktem Opisanym

Zarządzanie Punktem Własnym

Tworzenie trasy

Settings

Zgłoszenie do weryfikacji

Wybierz odcinki do weryfikacji

Trasa Lato 2021

LP.	POCZĄTEK	KONIEC	GRUPA GÓRSKA	LICZBA PUNKTÓW
1.	Wolosate	Przełęcz Bukowska	Bieszczady	8
2.	Przełęcz Bukowska	Rozsypaniec	Bieszczady	6
3.	Schronisko w Dolinie Roztoki	Wodogrzmoty Mickiewicza	Tatry	2
4.	Halicz	Przełęcz Goprowska	Bieszczady	8

Dalej

Dodane załączniki: 0
Przypisani weryfikujący: 0

Zgłoś



Zgłaszanie prośby o zweryfikowanie Trasy – alternatywny wybór odcinków

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

TURYSTA Z ODZNAKĄ

Wyszukaj

Jan Kowalski

Wybierz odcinki do weryfikacji

Trasa Lato 2021

Lp.	POCZĄTEK	KONIEC	GRUPA GÓRSKA	LICZBA PUNKTÓW
1.	Wolosate	Przełęcz Bukowska	Bieszczady	8
2.	Przełęcz Bukowska	Rozsypaniec	Bieszczady	6
3.	Schronisko w Dolinie Roztoki	Wodogrzmoty Mickiewicza	Tatry	2
4.	Halicz	Przełęcz Goprowska	Bieszczady	8

Nie można dodać załącznika dla odcinków z różnych grup górskich!

Dodaj załącznik Przypisz weryfikującego

Zgłoś Powrót

Zgłaszanie prośby o zweryfikowanie Trasy – błąd – chęć załączenia jednego dowodu dla odcinków z różnych grup górskich

9.1.4 Zarządzanie Punktem Własnym

TURYSTA Z ODZNAKĄ

Wyszukaj

Jan Kowalski

Wybierz, jaką akcję chciałbyś podjąć

Dodaj punkt własny Usuń punkt własny Edytuj punkt własny Szukaj punktu własnego

Zarządzanie Punktem Własnym – ekran startowy

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

TURYSTA Z ODZNAKĄ

Wyszukaj

Dodawanie punktu własnego

Nazwa punktu
Schronisko Dom Śląski

Szerokość geograficzna
50.74016245198827

Długość geograficzna
15.729052976702954

Dodaj punkt własny

Zarządzanie Punktem Własnym – dodawanie

TURYSTA Z ODZNAKĄ

Wyszukaj

Dodawanie punktu własnego

Nazwa punktu
Dom Śląski Schronisko

Wybrana nazwa już istnieje.

Szerokość geograficzna
50.74016245198827

Długość geograficzna
15.729052976702954

Dodaj punkt własny

Zarządzanie Punktem Własnym – dodawanie - zła nazwa (Naduszenie przycisku nie będzie możliwe dopóki użytkownik nie wprowadzi poprawnej nazwy i współrzędnych geograficznych).

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

Zarządzanie Punktem Własnym – dodawanie – zle współrzędne (Naduszenie przycisku nie będzie możliwe dopóki użytkownik nie wprowadzi poprawnej nazwy i współrzędnych geograficznych.

Zarządzanie Punktem Własnym – dodawanie - pomyślne dodanie punktu

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

TURYSTA Z ODZNAKĄ

- Strona główna
- Powiadomienia
- Zarządzanie Punktem Opisanym
- Zarządzanie Punktem Własnym
- Tworzenie trasy
- Settings

Wyszukaj

Usuwanie punktu własnego

Nazwa Punktu
Schronis

Schronisko Dom Śląski
Schronisko Samotnia
Schronisko Strzecha Akademicka

Usuń punkt własny



Zarządzanie Punktem Własnym – usuwanie

TURYSTA Z ODZNAKĄ

- Strona główna
- Powiadomienia
- Zarządzanie Punktem Opisanym
- Zarządzanie Punktem Własnym
- Tworzenie trasy
- Settings

Wyszukaj

Usuwanie punktu własnego

Nazwa Punktu
Schronisko w Kamieńcu

Punkt o wybranej nazwie nie istnieje

Usuń punkt własny



Zarządzanie Punktem Własnym – usuwanie – nieistniejący punkt (Naduszenie przycisku nie będzie możliwe dopóki użytkownik nie wprowadzi poprawnej nazwy)

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

TURYSTA Z ODZNAKĄ

Wyszukaj

Strona główna

Powiadomienia

Zarządzanie Punktem Opisanym

Zarządzanie Punktem Własnym

Tworzenie trasy

Settings

Usuwanie punktu opisanego

Nazwa Punktu
Schronisko przy Chełmiance

Punkty jest już używany w odcinkach. Nie można go usunąć.

Usuń punkt własny

Zarządzanie Punktem Własnym – usuwanie – już używany punkt (Naduszenie przycisku nie będzie możliwe dopóki użytkownik nie wprowadzi poprawnej nazwy)

TURYSTA Z ODZNAKĄ

Wyszukaj

Strona główna

Powiadomienia

Zarządzanie Punktem Opisanym

Zarządzanie Punktem Własnym

Tworzenie trasy

Settings

Usuwanie punktu własnego

Nazwa Punktu
Schronisko Dom Śląski

Usuń punkt własny

Zarządzanie Punktem Własnym – usuwanie

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

The screenshot shows the application's main menu on the left with options like 'Strona główna', 'Powiadomienia', 'Zarządzanie Punktem Opisanym', 'Zarządzanie Punktem Własnym', 'Tworzenie trasy', and 'Settings'. The central area has a search bar ('Wyszukaj') and a message 'Punkt został pomyślnie usunięty!' (Point was successfully deleted!). A large orange button labeled 'Zakończ' (Finish) is at the bottom right. Below the interface is a cartoon illustration of two hikers walking on a path through green hills under a blue sky.

Zarządzanie Punktem Własnym – usuwanie – pomyślne usunięcie punktu

The screenshot shows the application's main menu on the left. In the center, there's a section titled 'Edycja punktu własnego' (Edit own point) with a search bar for 'Nazwa Szukanego Punktu' (Name of the searched point). Below it are three options: 'Schronisk', 'Schronisko Dom Śląski', 'Schronisko przy Chełmiance', and 'Schronisko Strzecha Akademicka'. An orange button labeled 'Edytuj punkt własny' (Edit own point) is at the bottom right. Below the interface is a cartoon illustration of two hikers walking on a path through green hills under a blue sky.

Zarządzanie Punktem Własnym – edytowanie (wprowadzane dane sprawdzane są w identyczny sposób jak przy usuwaniu oraz dodawaniu nowego punktu)

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

TURYSTA Z ODZNAKĄ

Wyszukaj

Jan Kowalski

Edycja punktu własnego

Nazwa punktu
Schronisko przy Chełmiance

Szerokość geograficzna
50.74016245198827

Długość geograficzna
15.729052976702954

Edytuj punkt opisany

Zarządzanie Punktem Własnym – edytowanie – modyfikacja danych (wprowadzane dane sprawdzane są w identyczny sposób jak przy usuwaniu oraz dodawaniu nowego punktu)

TURYSTA Z ODZNAKĄ

Wyszukaj

Jan Kowalski

Punkt został pomyślnie edytowany!

Zakończ

Zarządzanie Punktem Własnym – edytowanie – pomyślna edycja punktu

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

TURYSTA Z ODZNAKĄ

Wyszukaj

Szukanie punktu własnego

Nazwa Szukanego Punktu
Przełęcz

Przełęcz Chocholowska
Przełęcz pod Chłopkiem
Kozia Przełęcz

Szukaj punktu własnego



Zarządzanie Punktem Własnym – szukanie

TURYSTA Z ODZNAKĄ

Wyszukaj

Szukanie punktu własnego

NAZWA	SZEROKOŚĆ GEOGRAFICZNA	DŁUGOŚĆ GEOGRAFICZNA
Przełęcz Chocholowska	118.74016245198827	15.729052976702954
Przełęcz pod Chłopkiem	118.74016245198827	15.729052976702954
Kozia Przełęcz	118.74016245198827	15.729052976702954

Powrót



Zarządzanie Punktem Własnym – szukanie – wyświetlanie wyników pasujących do frazy “Przełęcz”

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

Zarządzanie Punktem Własnym – szukanie – brak pasujących wyników wyszukiwania

9.1.5 Zarządzanie Punktem Opisanym

Dodawanie punktu - wątek główny

Zarządzanie punktem opisanym – start

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

Zarządzanie punktem opisanym - główny - dodanie punktu

Zarządzanie punktem opisanym - główny - koniec

Wątki alternatywne – plansze dla alternatywnych wątków

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

Zarządzanie punktem opisanym - dodanie punktu – złe dane (jeżeli punkt już istnieje)

Wysokość to atrybut opcjonalny – da się tylko wprowadzić dodatnie liczby całkowite
Przycisk “Dodaj punkt opisany” da się nacisnąć tylko, gdy dane w formularzu są poprawne
Po poprawieniu niepoprawnych danych, przeniesieni zostaniemy do kroku: Zarządzanie punktem opisanym - główny - koniec

Usuwanie punktu - wątek główny

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

Zarządzanie punktem opisanym - główny - usunięcie punktu

Zarządzanie punktem opisanym - główny - usunięcie punktu – dropdown list

Zarządzanie punktem opisanym - główny - usunięcie punktu – wybranie

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

The screenshot shows the application's main interface. On the left is a green sidebar with the title "TURYSTA Z ODZNAKĄ" and a navigation menu containing links: Strona główna, Powiadomienia, Zarządzanie Punktem Opisanym, Zarządzanie Punktem Własnym, Tworzenie trasy, and Settings. The main content area has a search bar at the top right. A success message "Punkt został pomyślnie usunięty!" (Point was successfully deleted!) is displayed above a large, stylized illustration of a hiker standing on a path in a mountainous landscape. Below the illustration is a button labeled "Zakończ" (Finish).

Zarządzanie punktem opisanym - główny - usunięcie punktu – koniec

Wątki alternatywne – plansze dla alternatywnych wątków

This screenshot shows a similar interface to the previous one, but with a different outcome. The main content area displays a form titled "Usuwanie punktu opisanego" (Delete marked point) with a field labeled "Nazwa Punktu" (Point name) containing the value "Watlina". A small note below the field states "Punkt o wybranej nazwie nie istnieje" (Point with the selected name does not exist). A button labeled "Usuń punkt opisany" (Delete marked point) is visible. The rest of the interface, including the sidebar and the hiker illustration, is identical to the first screenshot.

Zarządzanie punktem opisanym - usunięcie punktu - punkt nie istnieje

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

Zarządzanie punktem opisanym - usunięcie punktu - punkt jest już używany

W przypadku powyższych wątków alternatywnych, naduszenie przycisku “Usuń punkt opisany” będzie nemożliwe, dopóki podawane są niepoprawne dane lub wybrany punkt jest już używany np. w Odcinku Opisanym.
Po poprawnym wypełnieniu formularza, po naciśnięciu przycisku, przeniesieni zostaniemy do makiety: “Zarządzanie punktem opisanym - główny - usunięcie punktu – koniec”

Edytowanie punktu - wątek główny

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

Zarządzanie punktem opisanym - główny - edycja punktu

The screenshot shows the application's main menu on the left with options like 'Strona główna', 'Powiadomienia', 'Zarządzanie Punktem Opisanym', 'Zarządzanie Punktem Własnym', 'Tworzenie trasy', and 'Settings'. The central area is titled 'Usuwanie punktu opisanego' (Delete marked point) with a text input field containing 'Smerek'. A large orange button labeled 'Edytuj punkt opisany' (Edit marked point) is visible. Below the form is a cartoon illustration of a hiker standing on a path in a mountainous landscape.

Zarządzanie punktem opisanym - główny - edycja punktu – wybranie

The screenshot shows the application's main menu on the left. The central area is titled 'Edycja punktu opisanego' (Edit marked point) with a text input field containing 'Smerek'. Below it is another input field for 'Nowa Wysokość Punktu (w metrach)' (New point height in meters) with the value '1223'. A large orange button labeled 'Edytuj punkt opisany' (Edit marked point) is visible. Below the form is a cartoon illustration of a hiker standing on a path in a mountainous landscape.

Zarządzanie punktem opisanym - główny - edycja punktu - podawanie nowych danych

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

Zarządzanie punktem opisanym - główny - edycja punktu – koniec

Wątki alternatywne – plansze dla alternatywnych wątków

Zarządzanie punktem opisanym - edycja punktu – punkt nie istnieje

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

Zarządzanie punktem opisanym - edycja punktu – punkt jest już używany

W przypadku powyższych wątków alternatywnych, naduszenie przycisku “Edytuj punkt opisany” będzie nimożliwe, dopóki podawane są niepoprawne dane lub wybrany punkt jest już używany np. w Odcinku Opisanym.
Po poprawnym wypełnieniu formularza, po naciśnięciu przycisku, przeniesieni zostaniemy do makiety: “Zarządzanie punktem opisanym - główny - edycja punktu - podawanie nowych danych”

Zarządzanie punktem opisanym - edycja punktu – zle nowe dane - wprowadzanie złych nowych danych

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

W przypadku powyższego wątku alternatywnego, naduszenie przycisku “Edytuj punkt opisany” będzie nemożliwe, dopóki podawane są niepoprawne dane np. punkt o danej nazwie już istnieje.

Po poprawnym wypełnieniu formularza, po naciśnięciu przycisku, przeniesieni zostaniemy do makiety: “Zarządzanie punktem opisanym - główny - edycja punktu – koniec”

Szukanie punktu - wątek główny

Zarządzanie punktem opisanym - główny - szukanie punktu

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

The screenshot shows the main interface of the 'Turysta Z Odznaką' application. On the left is a green sidebar with the title 'TURYSTA Z ODZNAKĄ' and a list of menu items: Strona główna, Powiadomienia, Zarządzanie Punktem Opisanym (highlighted with a blue background), Zarządzanie Punktem Własnym, Tworzenie trasy, and Settings. The main content area has a search bar at the top with the placeholder 'Wyszukaj'. Below it is a search form titled 'Szukanie punktu opisanego' with a text input field containing 'Nazwa Szukanego Punktu' and the value 'Smer'. A large orange button labeled 'Szukaj punktu opisanego' is centered below the input field. In the bottom right corner of the main area, there is a decorative illustration of a person with a backpack standing on a path in a mountainous landscape.

Zarządzanie punktem opisanym - główny - szukanie punktu – wybranie

The screenshot shows the search results for the marked point 'Smer'. The search bar at the top now displays 'Szukanie punktu opisanego: Smer'. Below it is a table with two rows of results:

NAZWA	WYSOKOŚĆ (nie obowiązkowa)
Smerek	1222 m
Smerekowiec	-

In the bottom right corner of the main area, there is a decorative illustration of a person with a backpack standing on a path in a mountainous landscape.

Zarządzanie punktem opisanym - główny - szukanie punktu – wyświetlanie

Wątki alternatywne – plansze dla alternatywnych wątków

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

The screenshot shows the application's main interface. On the left is a green sidebar with the title "TURYSTA Z ODZNAKĄ" and a list of menu items: Strona główna, Powiadomienia, Zarządzanie Punktem Opisanym (which is highlighted in blue), Zarządzanie Punktem Własnym, Tworzenie trasy, and Settings. At the top right is a user profile for "Jan Kowalski". The main area has a search bar with the placeholder "Wyszukaj" and a button "Szukaj punktu opisanego". Below the search bar is a text input field with the placeholder "Nazwa Punktu" containing the text "Smirk". A large orange button labeled "Szukaj punktu opisanego" is centered below the search bar. To the right of the search area is a stylized illustration of a hiker standing on a path in a mountainous landscape.

Zarządzanie punktem opisanym - szukanie punktu – nieistniejący punkt

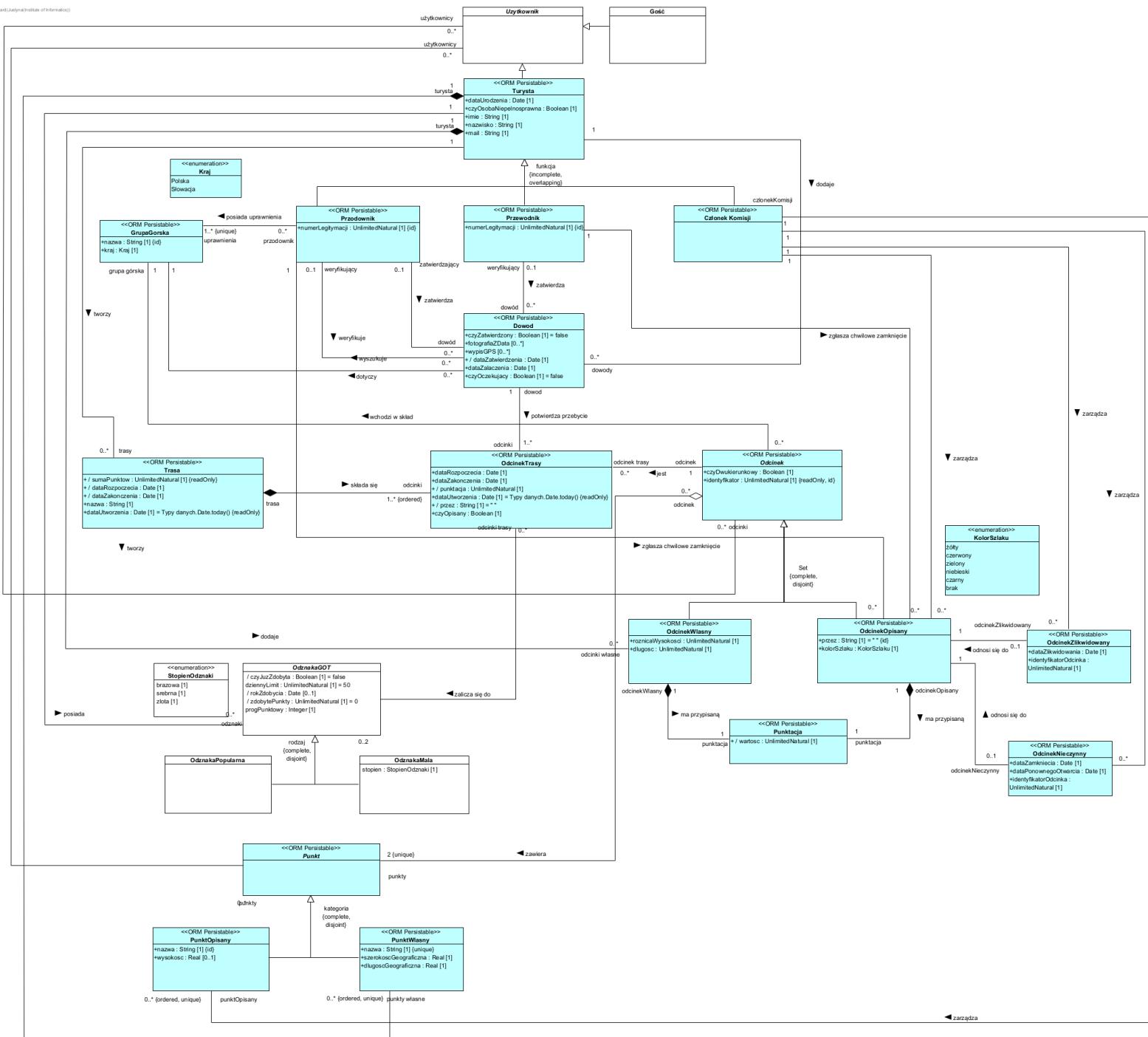
This screenshot shows the same application interface as the previous one, but the search results are different. The search bar contains "Smirk" and the search button is visible. Below the search bar, the text "Brak wyników" (No results) is displayed in orange. The rest of the interface, including the sidebar and the hiker illustration, remains the same.

Zarządzanie punktem opisanym - szukanie punktu – wyświetlanie – brak wyników

W przypadku powyższego wątku alternatywnego, w przypadku, gdy nie ma dopasowania w bazie, pojawi się komunikat: “Brak wyników”.

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

Model informacyjny



“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

REGUŁY BIZNESOWE W JĘZYKU OCL

Turysta musi mieć ukończony 7 rok życia

context Turysta

inv: (self.dataUrodzenia.getYear() + 7 > DateTime.Now.getYear()) or
 ((self.dataUrodzenia.getYear() + 7 = DateTime.Now.getYear()) and self.dataUrodzenia.getMonth() >
 DateTime.Now.getMonth()) or
 ((self.dataUrodzenia.getYear() + 7 = DateTime.Now.getYear()) and self.dataUrodzenia.getMonth() =
 DateTime.Now.getMonth() and
 (self.dataUrodzenia.getDay() >= DateTime.Now.getDay()))

Punkt musi być identyfikowany przez swoją nazwę

context PunktOpisany

inv: PunktOpisany.allInstances() -> forAll(p1, p2 | p1 <> p2 implies p1.nazwa <> p2.nazwa)

Punkt własny musi być identyfikowany przez współrzędne geograficzne. Wszystkie punkty własne turysty muszą posiadać unikalną nazwę

context Turysta

inv: self.punktywlasne -> forAll(p1, p2 | p1 <> p2 implies (p1.szerokoscGeograficzna <> p2.
 szerokoscGeograficzna or p1.dlugoscGeograficzna <> p2. dlugoscGeograficzna))

context Turysta

inv: self.punktywlasne -> forAll(p1, p2 | p1 <> p2 implies p1.nazwa <> p2.nazwa)

Przodownik musi posiadać numer legitymacji

context Przodownik

inv: self.numerLegitymacji > 0;

inv: Przodownik.allInstances() -> forAll(p1, p2 | p1 <> p2 implies p1.numerLegitymacji <> p2 numerLegitymacji)

Przewodnik musi posiadać numer legitymacji

context Przewodnik

inv: self.numerLegitymacji > 0;

inv: Przewodnik.allInstances() -> forAll(p1, p2 | p1 <> p2 implies p1.numerLegitymacji <> p2 numerLegitymacji)

Odcinek musi wchodzić w skład jednej grupy górskiej

context Odcinek

inv: self.grupagórska -> size() = 1

Nie można planować wycieczki przez dany odcinek, jeżeli jest on chwilowo nieczynny lub całkowicie zlikwidowany

context Trasa

inv: self.odcinki -> forAll(o | (o.odcinek.ocIsTypeOf(OdcinekOpisany) = false) or
 ((o.odcinek.ocIsTypeOf(OdcinekOpisany) = true) and
 (OdcinekZlikwidowany.allInstances() -> exist (oz | oz.identyfikatorOdcinka = o.odcinek.identyfikator) = false or
 (OdcinekZlikwidowany.allInstances() ->
 exist (oz | oz.identyfikatorOdcinka = o.odcienk.identyfikator and o.dataUtworzenia < oz.dataZlikwidowania))
) and
 (OdcinekNiezczynny.allInstances() -> exist (on | on.identyfikatorOdcinka = o.odcinek.identyfikator) = false or
 (OdcinekNiezczynny.allInstances() -> select (on | on.identyfikatorOdcinka = o.odcinek.identyfikator) ->
 forAll(on | o.dataUtworzenia < on.dataZamkniecia or o.dataUtworzenia > on.dataPonownegoOtwarcia)))

Turysta musi wprowadzić różnicę wysokości punktu początkowego i końcowego oraz długość dodawanego odcinka własnego

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

context OdcinekWlasny

inv: self.roznicaWysokosci <> null

inv: self.dlugosc <> null

W ciągu jednego roku kalendarzowego można zdobyć...

context Turysta

inv: self.odznaki -> forAll(o | (self.odznaki -> select(od | o.rokZdobycia = od.rokZdobycia) -> size() = 1) or (self.odznaki -> select(od | o.rokZdobycia = od.rokZdobycia and ((o.oclIsTypeOf(OdznakaPopularna) and od.oclIsTypeOf(OdznakaMala) and od.stopien = #brazowa) or (od.oclIsTypeOf(OdznakaPopularna) and o.oclIsTypeOf(OdznakaMala) and o.stopien = #brazowa))) -> size() = 2))

Turysta musi uzyskać odznakę popularną, jeżeli uzyskał on na jej poczet 60 punktów, a w przypadku niepełnosprawnych turystów - 45 punktów

context OdznakaPopularna:**inv**

inv: if self.zdobytePunkty >= self.progPunktowy then self.czyJuzZdobra = true
endif

Turysta musi uzyskać odznakę małą złotą, jeżeli...

context OdznakaMala:**inv**

inv: if self.stopien = #zlota and self.zdobytePunkty >= self.progPunktowy and self.turysta.odznaki -> exists (o | o.oclIsTypeOf(OdznakaMala) and o.stopien = #srebrna) then self.czyJuzZdobra = true
endif

Punktacja za trasę musi być obliczona jako suma punktów należytych za odcinki wchodzące w skład trasy

context Trasa::sumaPunktow: Integer

derive: self.odcinki -> iterate (punkty; acc: integer = 0 | acc + punkty)

Odcinek własny musi być powiązany z dokładnie jednym turystą.

context OdcinekWlasny

inv: self.turysta -> size() = 1

Dowód musi potwierdzać przebycie co najmniej jednego odcinka trasy w obrębie jednej grupy górskiej

context Dowod

inv: self.odcinki -> size() >= 1 and self.odcinki -> forAll(o1, o2 | o1.odcinek.grupaGorska.nazwa = o2.odcinek.grupaGorska.nazwa)

Nie można zaliczać punktów uzyskanych na tych samych odcinkach i w tym samym kierunku w toku zdobywania tego samego stopnia GOT PTTK

context Turysta

inv: self.odznaki -> forAll(o | o.odcinkiTrasy -> forAll(ot1, ot2 | ot1 <> ot2 implies ot1.identyfikator <> ot2.identyfikator))

Punktacja za odcinki własne musi być obliczana według następujących zasad:...

context OdcinekWlasny::punktacja: Integer

derive: (self.roznicaWysokosci/100).round() + (self.dlugosc/1000).round()

Po zmianie statusu odznaki na zdobytą, należy natychmiastowo przypisać rok zdobycia odznaki

context Odznaka::rokZdobycia: Date

init: self.rokZdobycia = null

derive: if self.czyJuzZdobra = true then DateTime.Now.getYear() endif

Po zdobyciu odpowiedniej ilości punktów, należy natychmiastowo zmienić status odznaki na zdobytą.

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

context Odznaka::czyJuzZdobyta: Boolean
init: self.czyJuzZdobyta = False
derive: if self.zdobytePunkty >= self.progPunktowy then True endif

Nadwyżka punktów na poczet następnego małego stopnia odznaki musi być obliczona jako 50% normy potrzebnej do jej zdobycia

context Odznaka **inv:**

```
if self.oclIsTypeOf(OdznakaMala) and self.stopien = #brazowa then
let poprzedniaOdznaka: Odznaka = self.turysta.odznaki -> select(o | o.oclIsTypeOf(OdznakaPopularna)) -> first() in
self.zdobytePunkty = self.zdobytePunkty + (poprzedniaOdznaka.zdobytePunkty -
poprzedniaOdznaka.progPunktowy).min(0.5*poprzedniaOdznaka.progPunktowy)
else if self.oclIsTypeOf(OdznakaMala) and self.stopien = #srebrna then
let poprzedniaOdznaka: Odznaka = self.turysta.odznaki -> select(o | o.oclIsTypeOf(OdznakaMala) and o.stopien =
#brazowa) -> first() in self.zdobytePunkty = self.zdobytePunkty + (poprzedniaOdznaka.zdobytePunkty -
poprzedniaOdznaka.progPunktowy).min(0.5*poprzedniaOdznaka.progPunktowy)
else if self.oclIsTypeOf(OdznakaMala) and self.stopien = #zlota then
let poprzedniaOdznaka: Odznaka = self.turysta.odznaki -> select(o | o.oclIsTypeOf(OdznakaMala) and o.stopien =
#srebrna) -> first() in self.zdobytePunkty = self.zdobytePunkty + (poprzedniaOdznaka.zdobytePunkty -
poprzedniaOdznaka.progPunktowy).min(0.5*poprzedniaOdznaka.progPunktowy) endif
```

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

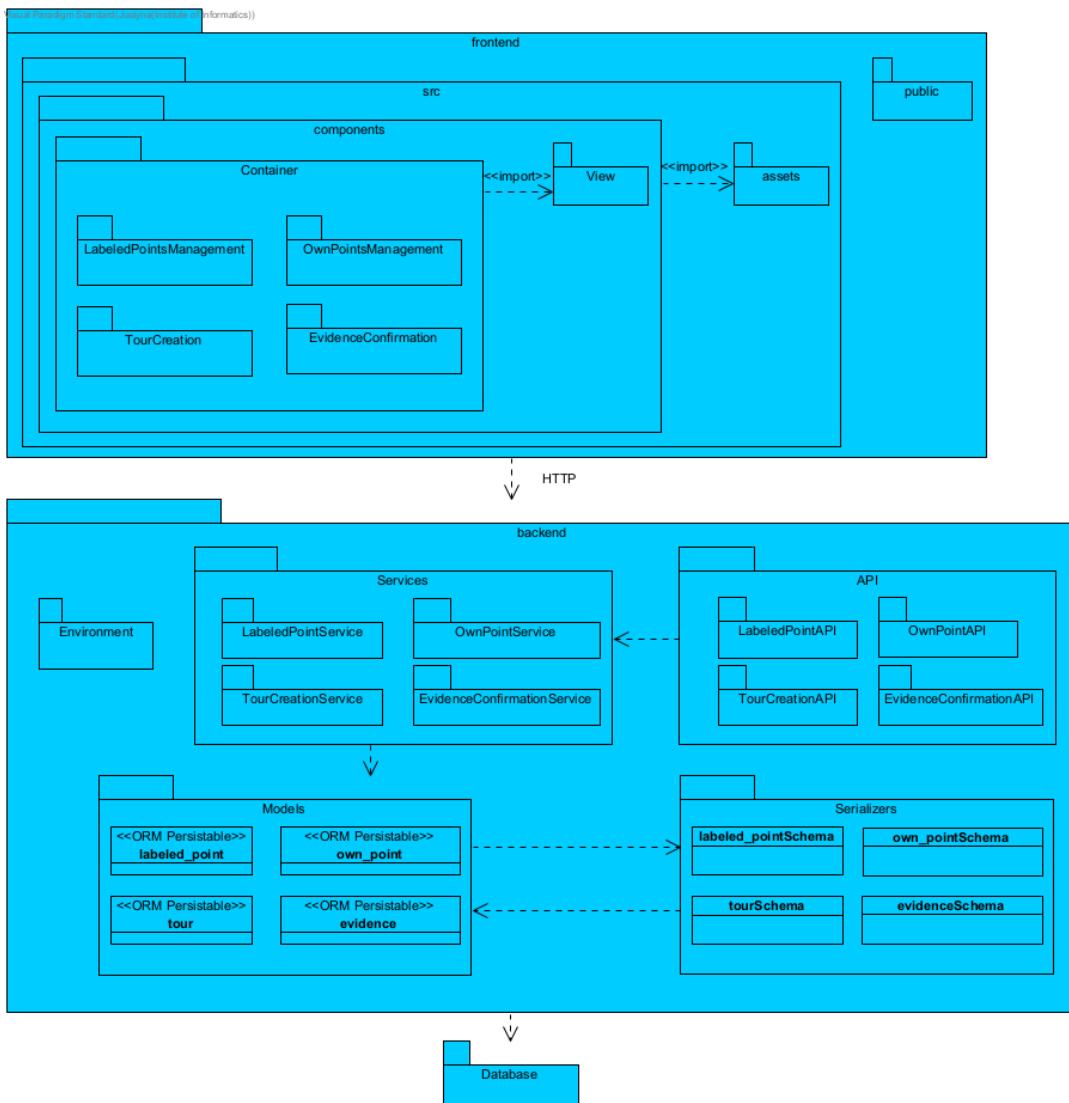
Etap III

Zmiany wprowadzone do etapu II:

- Zmiana liczności datyZatwierdzenia w Dowodzie na [0..1]
- Zmiana liczności datyPonownegoOtwarcia w Odcinku Nieczynnym na [0..1]
- Zmiana liczności datyRozpoczecia i datyZakonczenia w OdcinkuTrasy na [0..1]
- Zmiana liczności asocjacji Dowodu po stronie Dowodu na [0..1]
- Zmiana liczności datyRozpoczecia i datyZakonczenia Trasy na [0..1]
- Zmiana liczności atrybutu “przez” w odcinku opisany na [0..1]
- Poprawienie drobnych błędów w diagramach aktywności zarządzania punktem opisanym i tworzenia trasy

Architektura oprogramowania

1. Architektura logiczna



“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

Aplikacja została zaprojektowana z wykorzystaniem wzorca warstwowego. Podzielona jest na trzy warstwy: warstwę prezentacji, warstwę dziedzinową, logiki biznesowej i dostępu do danych oraz bazę danych.

Warstwa prezentacji zostanie wykonana przy pomocy biblioteki React i narzędzia [Create React App](#) oraz umieszczona w pakiecie frontend. Wyżej wymienione narzędzie tworzy strukturę projektu wraz z najważniejszymi zależnościami i konfiguracjami niezbędnymi do tworzenia projektu w React. Narzuca też istnienie pakietów: src (kod źródłowy) oraz public (plik index.html oraz kilka innych). W pakiecie frontend znajdują się jeszcze będą pliki konfiguracyjne. Pakiet src podzielony jest na dwa pakiety: assets, zawierający pliki graficzne oraz components, zawierający kod źródłowy komponentów.

W pakiecie components zastosowany zostanie wzorzec Container View, który ma na celu podzielenie komponentów na komponenty czysto prezentacyjne (View) oraz komponenty zarządzające stanem, wykonujące prostą logikę, przetwarzające dane (Container).

Dodatkowo, pakiet container zawiera w sobie kolejne cztery pakiety, jeden na każdy przypadek użycia, który należy zaimplementować. Jest to spowodowane tym, iż dobre praktyki programowania w React sugerują konwencję: jeden komponent, to jeden plik. Z tego względu plików składających się na warstwę prezentacji będzie bardzo dużo. Pogrupowanie tych komponentów w pakiety ma ułatwić developerom proces twórczy.

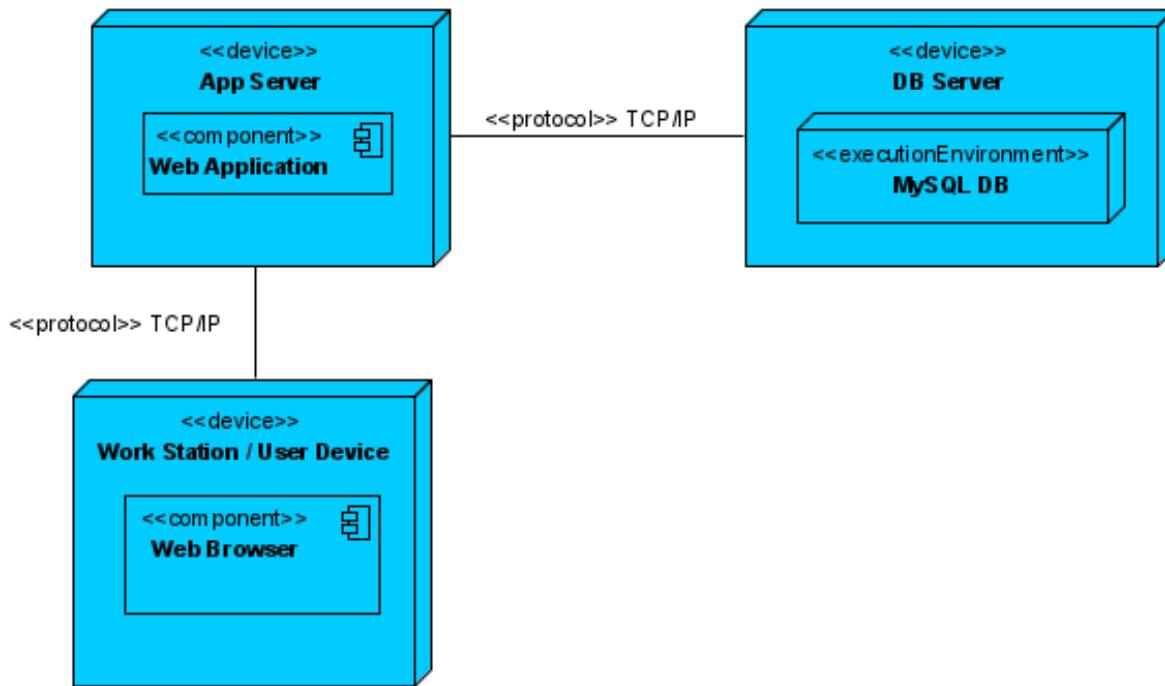
Warstwa dziedzinowa, logiki biznesowej i dostępu do danych obsługiwana będzie za pomocą technologii Python ze szczególnym wykorzystaniem biblioteki Flask. Cała warstwa znajdująca się będzie w pakiecie backend. Sam pakiet backend podzielony będzie na:

- Pakiet API – służący do komunikacji pomiędzy warstwą prezentacji a warstwą dziedzinową – wykorzystujący do tego celu zapytania REST API. Cała struktura REST'owa zostanie zaprojektowana przy użyciu wcześniej wspomnianej biblioteki Flask.
- Pakiet Models – pakiet wykorzystujący bibliotekę ORM – w naszym przypadku SQLAlchemy, która pozwala na zapytania i manipulowanie danymi z bazy danych przy użyciu paradymatu obiektowego. Zapewnia nam więc na połączenie z warstwą bazy danych. W pakiecie Models występuwać będą wszystkie potrzebne do realizacji przypadków użycia obiekty z modelu domenowego.
- Pakiet Serializers – pakiet zapewniający serializację i deserializację obiektów z pakietu Models na/do formatu JSON. Ułatwi to komunikację na poziomie zapytań REST'owych – Frontend otrzymywał będzie informację w formacie JSON. Backend w łatwy sposób przekonwertuje format JSON do obiektu Models.
- Pakiet Services – pakiet odpowiadający za całą logikę systemu.
- Pakiet Environment – pakiet służący do konfiguracji np. połączenia do bazy danych.

Systemem zarządzania bazą danych jest MySQL. Baza danych jest zatem relacyjną bazą danych. Silnikiem bazy jest InnoDB.

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

2. Architektura fizyczna



Na serwerze, na którym znajduje się aplikacja, wykonywana będzie cała logika biznesowa i przygotowywanie widoków dla użytkownika oraz realizowane będzie połączenie z bazą danych.

Z uwagi na to, że program jest aplikacją webową, komunikacja między użytkownikiem, a serwerem aplikacji odbywać się będzie poprzez protokół komunikacyjny TCP/IP. Podobnie, serwer, na którym znajduje się aplikacja webowa, będzie komunikował się z serwerem bazy danych poprzez ten sam protokół komunikacyjny.

Aplikacja webowa znajdująca się na serwerze App Server korzysta dodatkowo z biblioteki zewnętrznej React (ta z kolei korzysta z takich narzędzi jak Babel oraz Webpack oraz wiele innych) oraz frameworka Flask.

Baza danych nie jest rozproszona. Systemem zarządzania relacyjną bazą danych jest MySQL.

Urządzeniem użytkownika może być dowolny sprzęt z zainstalowaną przeglądarką, posiadający dostęp do internetu.

W przypadku naszego projektu, serwerami będą nasze stacje robocze z systemem operacyjnym Windows 10.

3. Model danych

W modelu danych zmianie uległy jedynie liczności niektórych atrybutów (wymienionych na początku dokumentacji Etapu 3). Poza zmianami liczbymi, nie nastąpiły żadne korekty. Model bazy danych został zbudowany 1:1 w stosunku do wcześniej przedstawionego modelu danych, dlatego nie załączamy go ponownie w tym miejscu.

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

4. Realizacja przypadków użycia

4.1.1 Zarządzanie Punktami Opisanym (Klaudia Błażyczek)

Na diagramie sekwencji, w części backendowej nie umieszczono modeli i serializerów, ponieważ z reguły czarnej skrzynki nie ingerujemy w sposób ich działania. Ponadto, z uwagi na to, że React jest deklaratywny, akcje, opisywane przez komunikaty, są częściowo napisane w języku naturalnym.

W diagramie sekwencji, zostały wprowadzone też zmiany w stosunku do diagramu aktywności.

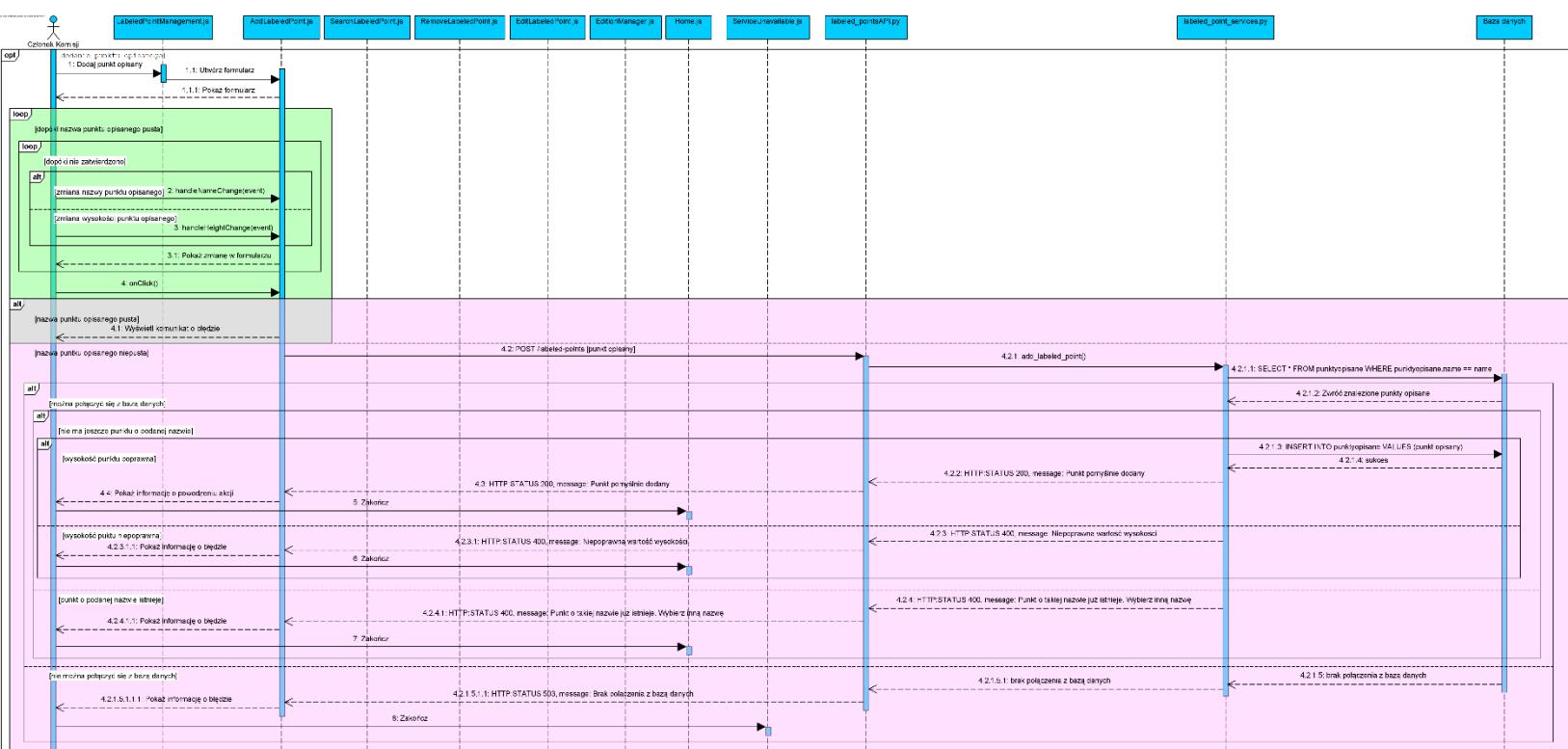
W przypadku dodawania punktu do bazy nie występuje w ogóle wczytanie punktów opisanych z bazy danych.

W przypadku odwołań do API, punkty opisane będą zawierać informację na temat odcinków, w których skład wchodzą. Dzięki temu, możliwa jest weryfikacja tego, czy punkt jest używany i czy można/nie można go edytować lub usunąć. Taki zabieg zimniejsza też ilość odwołań do bazy danych.

Dodatkowo, wprowadzono obsługę błędów związanych z łącznością z bazą danych.

W przypadku niektórych błędów popełnionych przez użytkownika (na przykład próba usunięcia punktu, który jest używany w odcinkach), zostaje wyświetlony komunikat i aktywność dobiera końca. Nie przeszkadza to jednak w tym, by użytkownik ponownie rozpoczął jedną z akcji zarządzania punktem opisanym.

Ze względu na rozmiar diagramu, został on podzielony w pliku dokumentacji na 3 części (dodawanie punktu, szukanie punktów i usuwanie punktu oraz edycja punktu).

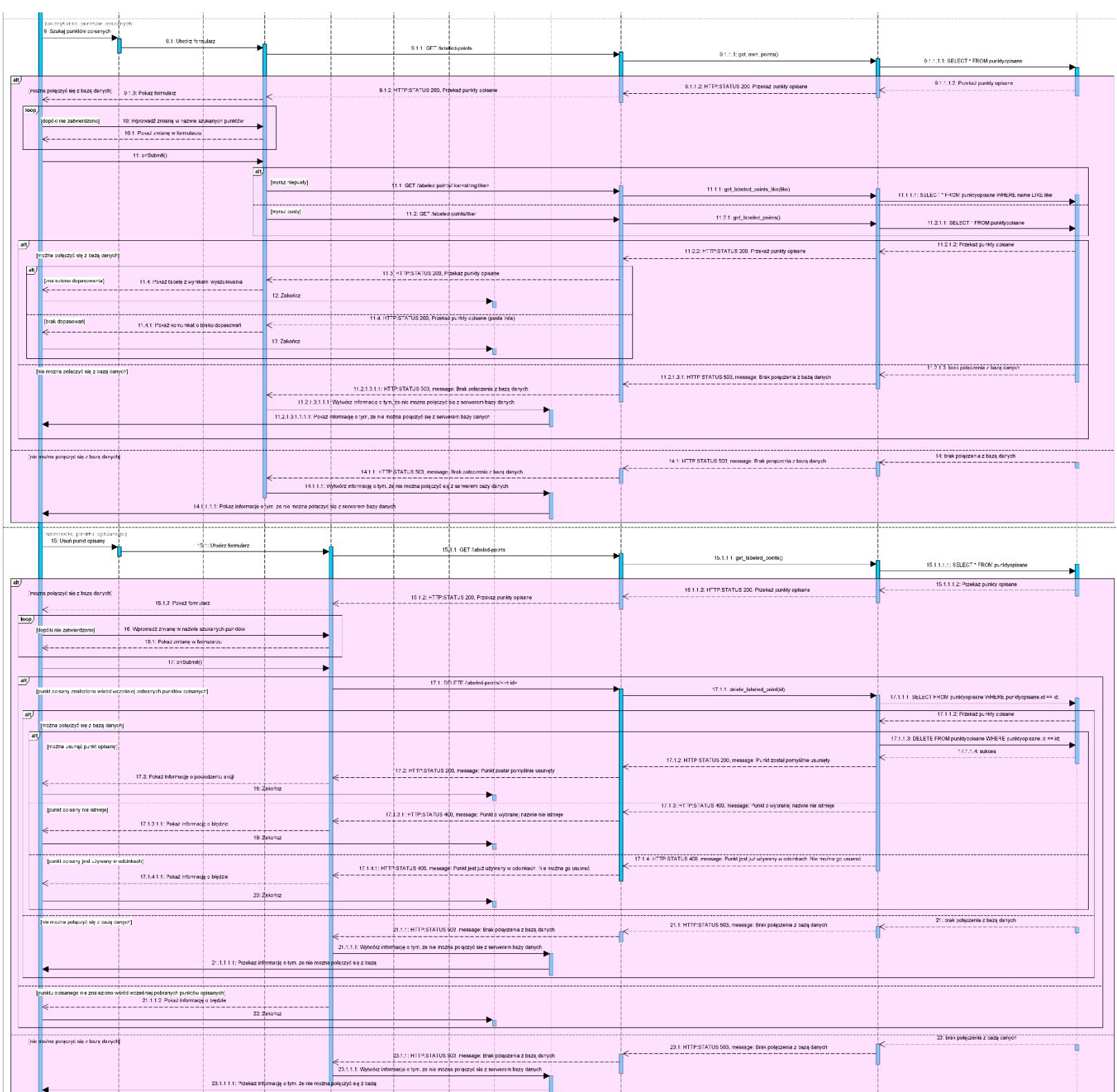


DODAWANIE - W momencie wybrania przez użytkownika akcji dodawania punktu, wyświetlany jest formularz, który na bieżąco przetwarza nowo wprowadzane dane po stronie frontendu. W chwili, gdy wypełnione są wszystkie pola obligatoryjne (nazwa) użytkownik może zapisać punkt. Wysyłany jest wtedy request POST do backendowej części systemu. Request jest odpowiednio weryfikowany (poprawność danych, połączenie z bazą danych). Jeśli nie występują błędy, do użytkownika zwracana jest informacja potwierdzająca dodanie punktu (kod 200). W przypadku wystąpienia błędu (połączenie z bazą, niepoprawność danych), zwracany jest stosowny komunikat informujący o błędzie (kod 400).

“Turysta Z Odznaką”<Project Name>

Etap I

Data: <dd/mmm/yy>



“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

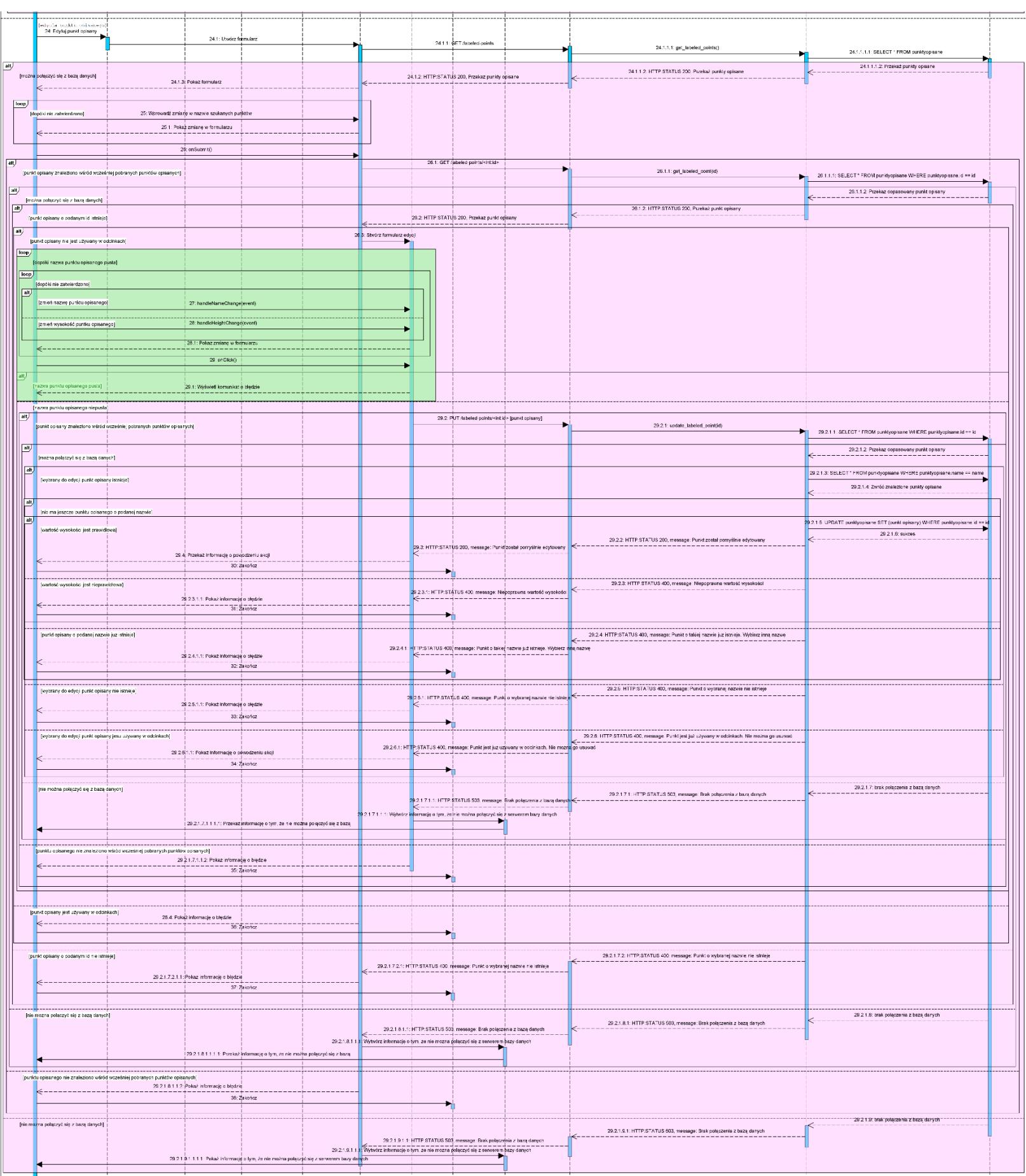
SZUKANIE – W chwili wybrania przez użytkownika opcji szukania punktu – wysyłany jest request z frontendu do backendu GET zwracający wszystkie punkty opisane. Służą one do “podpowiadania” użytkownikowi nazw podczas wpisywania ich w formularzu. Na tym etapie sprawdzamy też połączenie z bazą danych (czy jest w porządku). Jeżeli szukany term jest niepusty, wysyłany jest request GET, zwracający wszystkie punkty opisane, w których szukany term występuje w nazwie. Jeżeli szukany term jest pusty, wysyłany jest request GET, zwracający wszystkie punkty opisane. Jeśli zwracana lista punktów spełniająca wyżej zadane kryteria jest pusta, użytkownik jest informowany o braku dopasowań. Za każdym wysyłaniem zapytania do bazy danych, sprawdzana jest stabilność połączenia.

USUWANIE – Na samym początku wysyłane jest żądanie GET do backendu o zwrócenie wszystkich punktów opisanych. Służą one do podpowiadania użytkownikowi nazwy punktów opisanych w formularzu. Każdy wpisywany znak w formularzu jest na bieżąco przetwarzany przez frontend. W momencie kliknięcia przycisku usunięcia, na frontendzie weryfikowane jest czy usuwany punkt występował we wcześniej pobranej liście punktów. Jeżeli tak, to wysyłany jest request DELETE do warstwy backendowej. Jeżeli nie, to wyświetlany jest stosowny komunikat dla użytkownika. Request DELETE jest odpowiednio przetwarzany i weryfikowany (czy punkt istnieje, czy może być usunięty). Ponadto, standardowo weryfikowana jest stabilność połączenia z bazą danych.

“Turysta Z Odznaka”<Project Name>

Etap I

Data: <dd/mmm/yy>



“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

EDYTOWANIE – Podobnie jak w przypadku usuwania lub szukania, na początku pobierana jest lista punktów opisanych (oraz weryfikowana stabilność połączenia). Wykorzystując wcześniej już nam znany „system podpowiedzi nazw punktów”, oczekujemy i na bieżąco przetwarzamy wprowadzaną nazwę punktu. W momencie kliknięcia na przycisk potwierdzający edycję wybranego punktu, sprawdzamy czy jest on obecny we wcześniej pobranej liście i wysyłamy Request GET, podając dodatkowo id wybranego punktu. Weryfikowane jest też, czy punkt może zostać edytowany (punkty wchodzące w skład odcinków nie mogą być usunięte). Request odpowiednio przetworzony zwraca nam wybrany punkt (tylko gdy połączenie z bazą jest stabilne). Użytkownikowi wyświetlany jest formularz do edycji (wypełniony danymi punktu). Formularz w czasie rzeczywistym jest przetwarzany poprzez frontend. Poprzez naciśnięcie przycisku edycji, weryfikowane jest czy pole obligatoryjne nie jest puste (nazwa). Jeżeli jest niepuste, wysyłany jest Request PUT przekazujący nowe dane punktu. Dane te są odpowiednio weryfikowane (unikalność kluczowych atrybutów, czy istnieje). Jeżeli weryfikacja przebiegnie pomyślnie – zwracane jest potwierdzenie edycji punktu (kod 200). W przeciwnym wypadku zwracany jest kod błędu 400 z odpowiednim komunikatem.

4.1.2 Tworzenie Trasy (Klaudia Błażyczek)

Na diagramie sekwencji, w części backendowej nie umieszczono modeli i serializerów, ponieważ z reguły czarnej skrzynki nie ingerujemy w sposób ich działania. Ponadto, z uwagi na to, że React jest deklaratywny, akcje, opisywane przez komunikaty, są częściowo napisane w języku naturalnym.

W diagramie sekwencji, zostały wprowadzone też zmiany w stosunku do diagramu aktywności.

Wprowadzona została obsługa błędów związanych z łącznością z bazą danych.

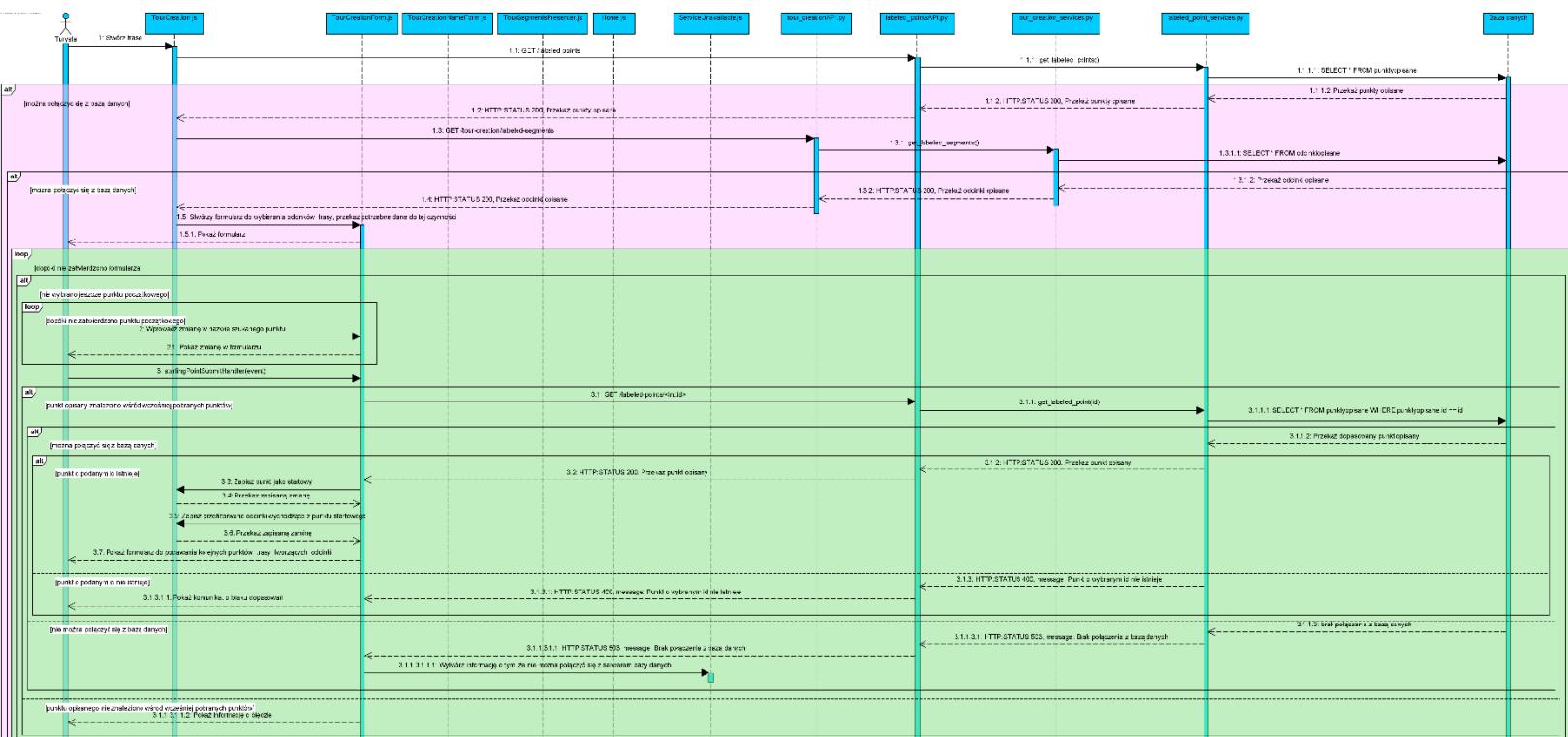
Niestety, ze względu na rozmiar i poziom skomplikowania diagramu, jedną rzeczą należy dopowiedzieć ustnie. W przypadku, gdy nie ma połączenia z bazą danych (komunikat nr. 3.1.1.3), po wyświetleniu informacji o błędzie, następuje koniec przypadku użycia – przerwanie pętli loop. Ze względu na położenie przypadków alternatywnych, nie dało pokazać tego na diagramie.

Ze względu na zmianę koncepcji działania przypadku użycia, edytowanie ostatniego punktu w trasie nie występuje. Można natomiast usuwać ostatni punkt w trasie i dodać w jego miejsce inny.

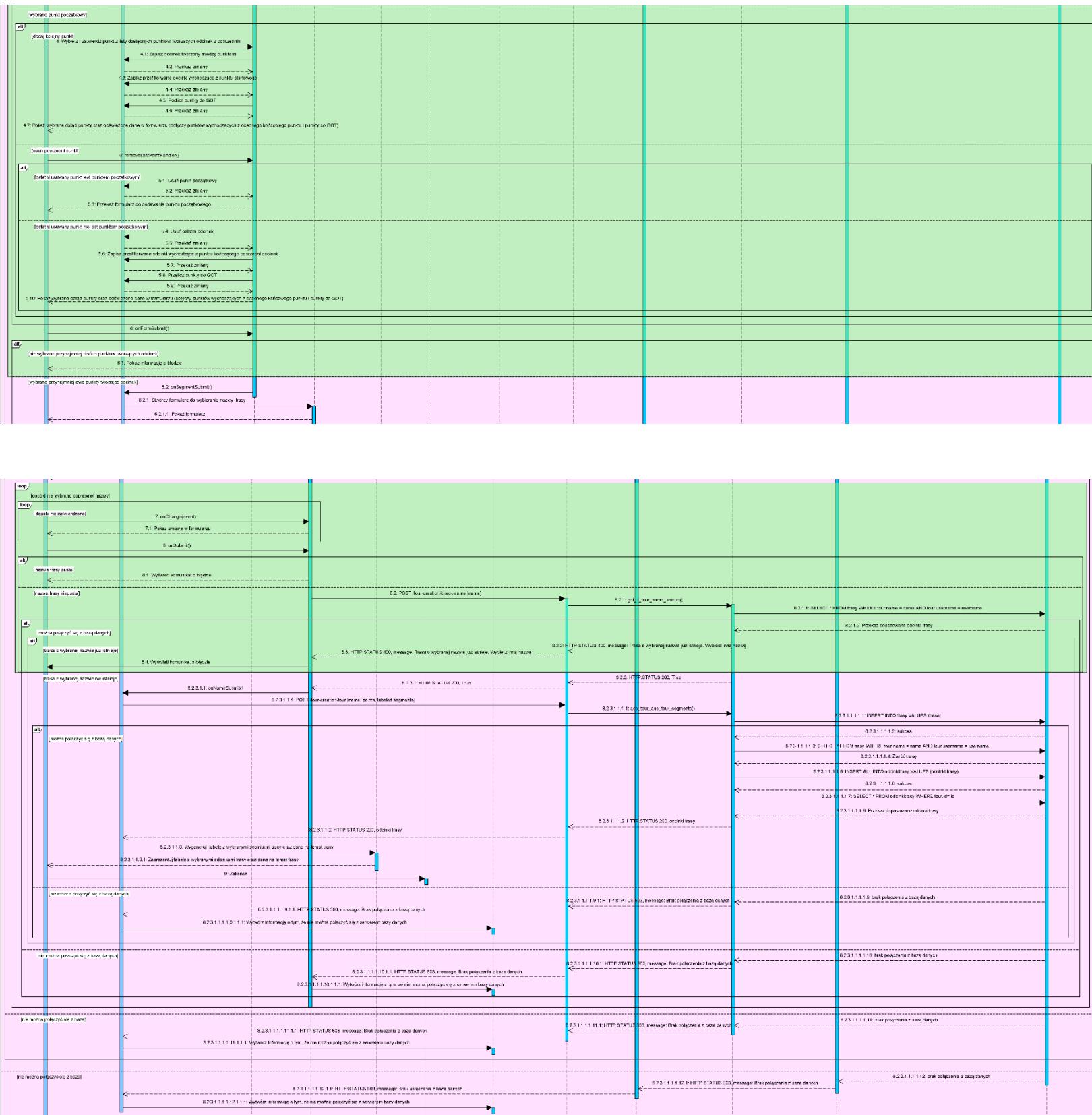
Przy nadawaniu nazwy trasie każdorazowo po zatwierdzeniu przez turystę, następuje odwołanie do backendu i bazy danych (nie dla nazwy pustej), by zweryfikować unikatowość nazwy trasy użytkownika.

Tworzenie w bazie trasy oraz odcinków trasy następuje na samym końcu, po zatwierdzeniu poprawnej nazwy.

Dla zwiększenia czytelności, diagram sekwencji w pliku z dokumentacją, został podzielony na trzy części, ze względu na swoją wielkość.



“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

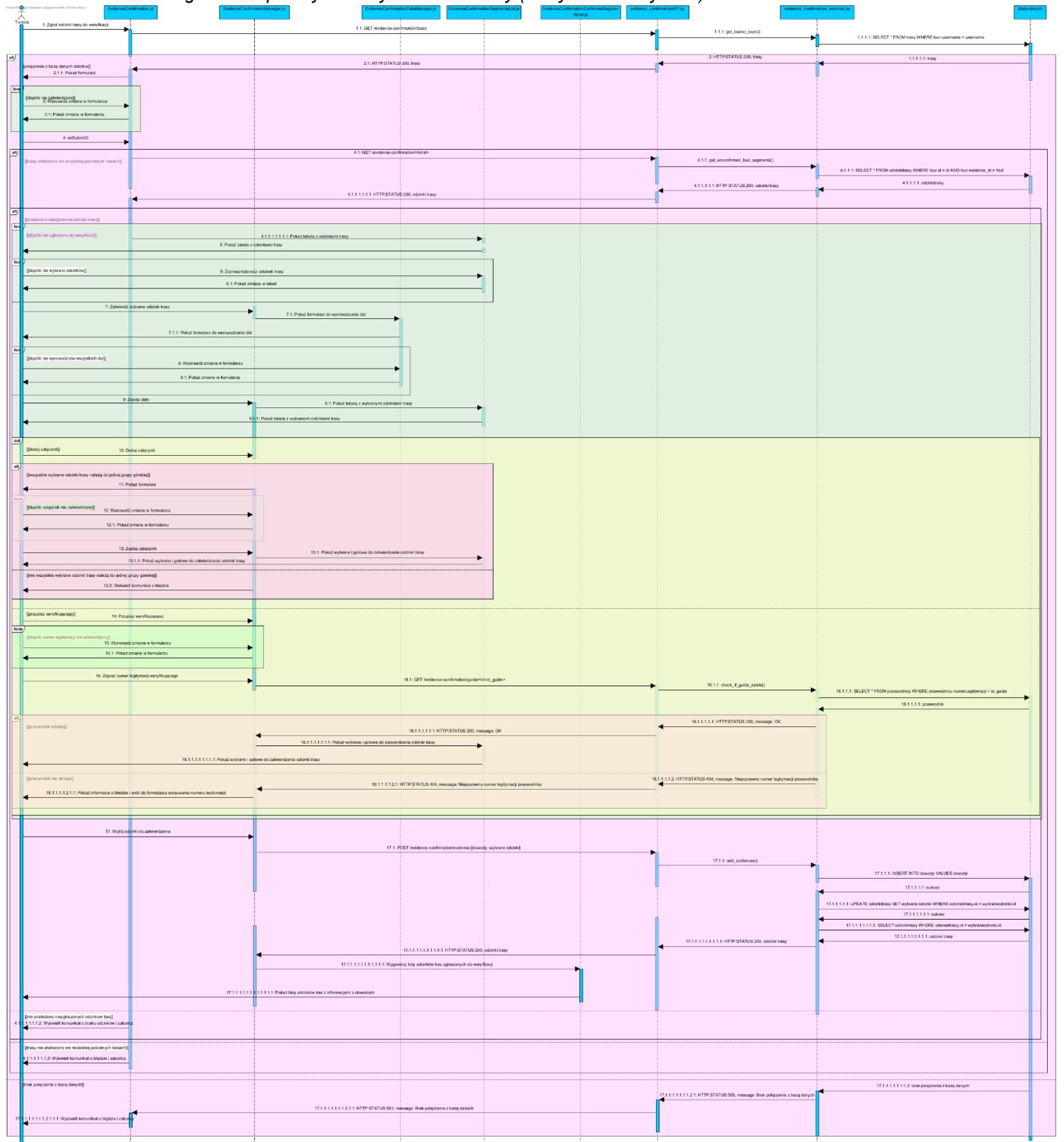


“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

Na początku pobierane są: lista punktów opisanych oraz lista odcinków opisanych. W dalszej części posłużą one do podpowiadania nazwy punktu początkowego oraz do wybierania kolejnych punktów, które będą tworzyć spójne wewspółnych punktach odcinki. Następnie turysta wpisuje punkt początkowy. Po zatwierdzeniu, poprawność punktu jest sprawdzana zarówno na frontendzie, jak i backendzie. Dopiero po podaniu poprawnego, istniejącego punktu początkowego, można przejść do dalszej części. W następnym kroku można dodawać kolejne punkty do trasy lub usuwać ostatnie punkty z trasy. Aplikacja po stronie frontendowej zapewnia ciągłość odcinków stworzonych z punktów. W przypadku, gdy użytkownik usunie wszystkie wybrane wcześniej punkty, powróci do formularza wybierającego punkt początkowy. Po zatwierdzeniu, pojawia się formularz do podania nazwy trasy. Turysta będzie proszony o podanie nazwy tak długo, dopóki nazwa jest pusta, lub turysta ma już trasę o podanej nazwie (odwołanie do API). Po pomyślnym wybraniu nazwy, aplikacja zapisuje trasę oraz odcinki trasy w bazie danych. Na samym końcu wyświetli tabelę z wybranymi odcinkami, nazwę trasy oraz liczbę punktów do GOT. Warto dodać, że w trakcie wybierania punktów trasy, na ekranie na bieżąco pojawia się suma punktów za trasę złożoną z wybranych na tamten moment odcinków. W komunikacji frontend – backend korzystamy z kodów błędów: 200 – akcja wykonana poprawnie, 400 – błąd po stronie użytkownika, 503 – brak połączenia z bazą danych. Komunikacja odbywa się poprzez protokół HTTP.

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

4.1.3 Zgłaszanie prośby o zweryfikowanie Trasy (Justyna Małuszyńska)

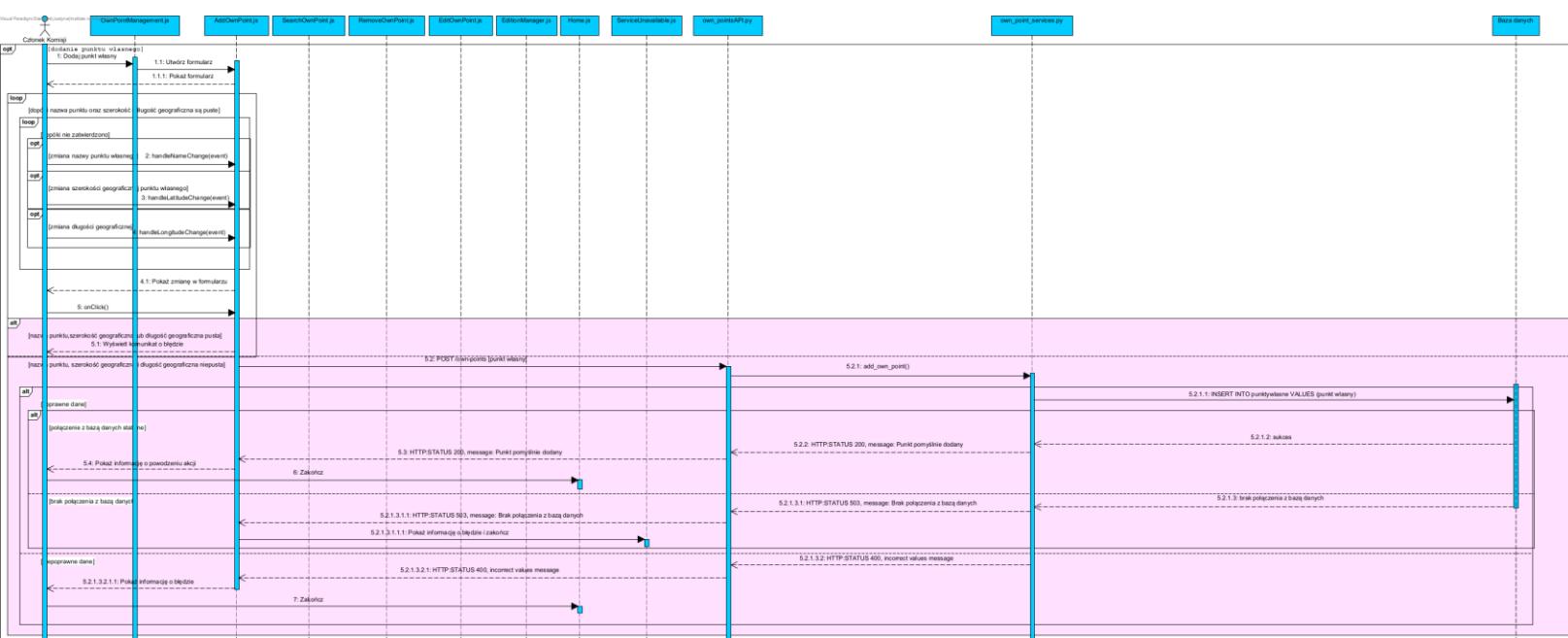


“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

Na początku pobierane z bazy są wszystkie trasy należące do zalogowanego użytkownika. Posłużą one w dalszym etapie do podpowiadania nazwy trasy użytkownikowi. System pokazuje formularz do wprowadzenia interesującej użytkownika trasy. Jeśli wprowadzona trasa znalazła się we wcześniej pobranych wszystkich trasach, wykonywane jest żądanie GET, które w odpowiedzi zwraca wszystkie niezgłoszone odcinki danej trasy. Następnie użytkownik ma możliwość wybrania konkretnych odcinków trasy dla których chce dodać wspólny dowód. Kolejnym krokiem jest umożliwienie wprowadzenia dat przebycia wybranych wcześniej odcinków. W momencie, gdy użytkownik chce zapisać daty, system umożliwia mu dwie opcje: wrzucenie dowodu w formie zdjęcia/pliku itd. lub przypisanie konkretnego przewodnika, który wybrane odcinki z nim przebył. Przewodnik jest dodatkowo weryfikowany odpowiednim żądaniem GET zwracającym informację o poprawności wpisanego numeru legitymacji. Czynność wybierania odcinków jest powtarzana dopóki nie skończą się odcinki lub dopóki użytkownik nie naciśnie przycisku wysłania do zatwierdzenia. Wysyłane jest żądanie POST, które przekazuje na stronę backendową informację o wybranych odcinkach, detach i formach wybranych dowodów. Wszystkie te informacje są przetwarzane w pakiecie services i zapisywane do bazy. Jako potwierdzenie zwracane są użytkownikowi dane odcinków oraz świeżo zapisane do nich dane.

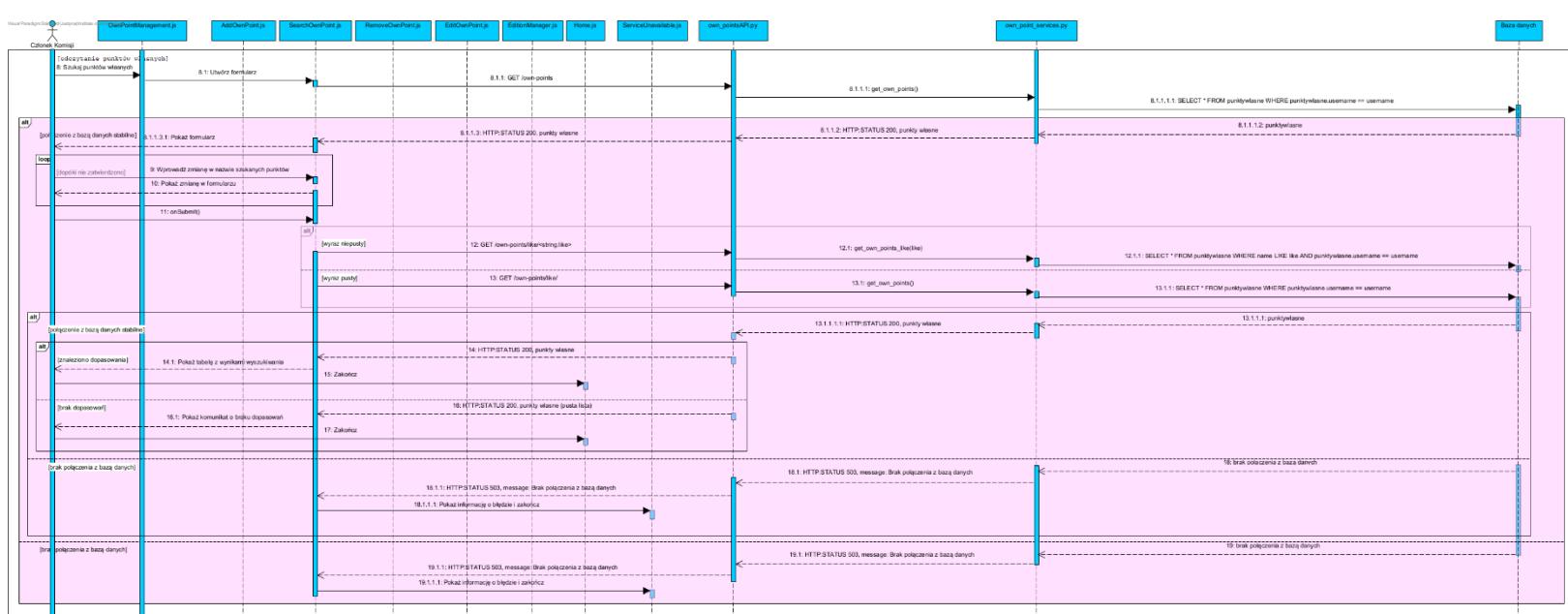
4.1.4 Zarządzanie Punktem Własnym (Justyna Małuszyńska)

Dla zwiększenia czytelności diagramu sekwencji, został on podzielony na 4 części (kolejno): dodawanie punktu, szukanie punktów, usuwanie punktu, edycja punktu.



DODAWANIE - W momencie wybrania przez użytkownika akcji dodawania punktu, wyświetlany jest formularz, który na bieżąco przetwarza nowo wprowadzane dane po stronie frontendu. W chwili, gdy wypełnione są wszystkie pola obligatoryjne (nazwa, szerokość i długość geograficzna) – wysyłany jest request POST do strony backendowej systemu. Request jest odpowiednio weryfikowany (poprawność danych, połączenie z bazą danych). Jeśli nie występują błędy, do użytkownika zwracana jest informacja potwierdzająca dodanie punktu (kod 200). W przypadku wystąpienia błędu (połączenie z bazą, niepoprawność danych), zwracany jest stosowny komunikat informujący o błędzie (kod 400).

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

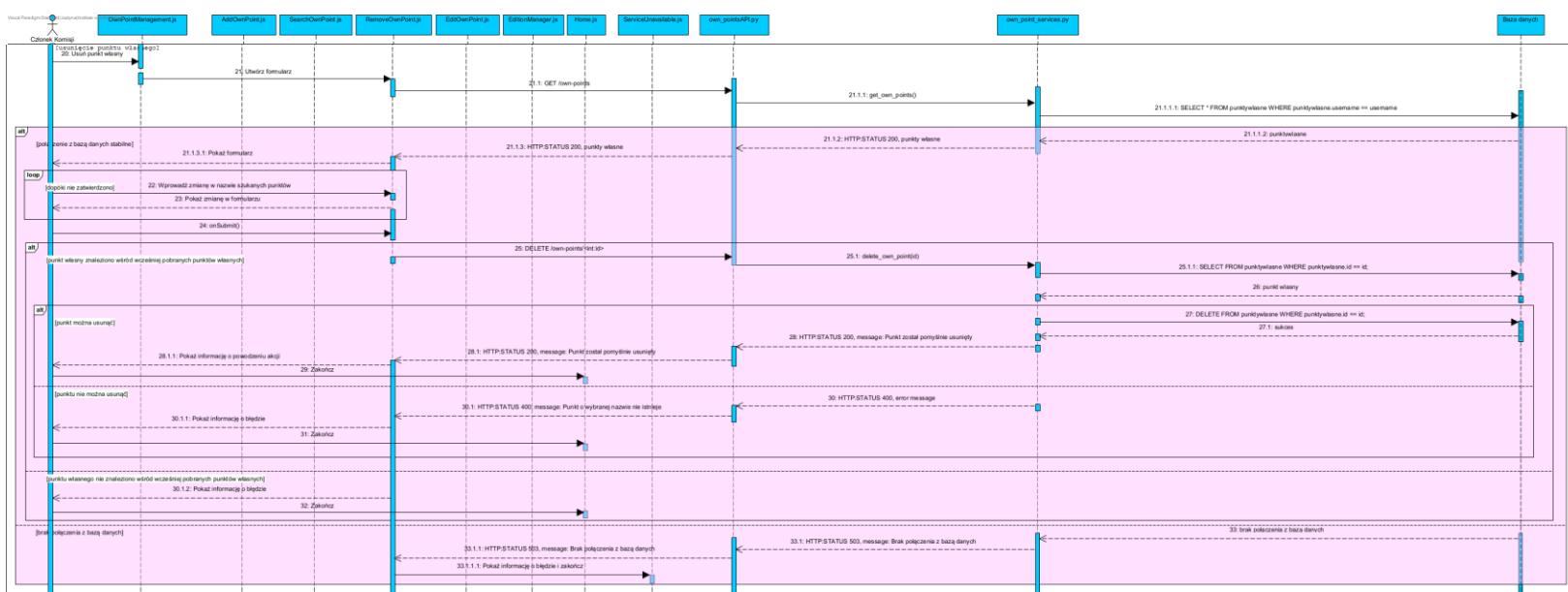


SZUKANIE – W chwili wybrania przez użytkownika opcji szukania punktu – wysyłany jest request z frontendu do backendu GET zwracający wszystkie punkty własne użytkownika. Służą one do “podpowiadania” użytkownikowi nazw podczas wpisywania ich w formularzu. Na tym etapie weryfikujemy stabilność połączenia z bazą danych. Jeżeli szukany term jest niepusty, wysyłany jest request GET, zwracający wszystkie punkty, w których szukany term występuje w nazwie. Jeżeli szukany term jest pusty, wysyłany jest request GET, zwracający wszystkie punkty własne użytkownika. Jeżeli zwracana lista punktów spełniająca wyżej zadane kryteria jest pusta, użytkownik jest informowany o braku dopasowań. Za każdym wysyłaniem zapytania do bazy danych, sprawdzana jest stabilność połączenia.

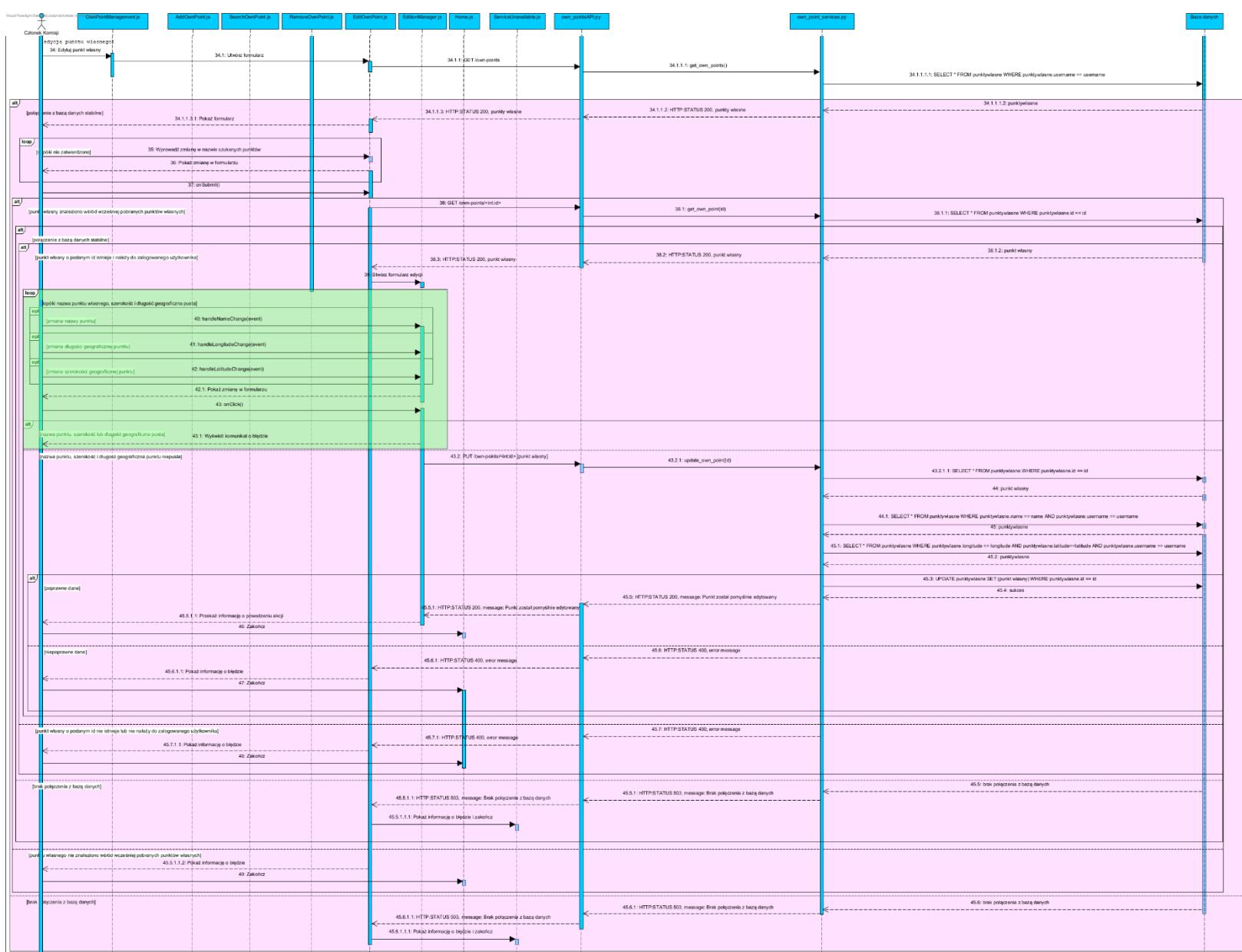
“Turysta Z Odznaką”<Project Name>

Etap I

Data: <dd/mmm/yy>



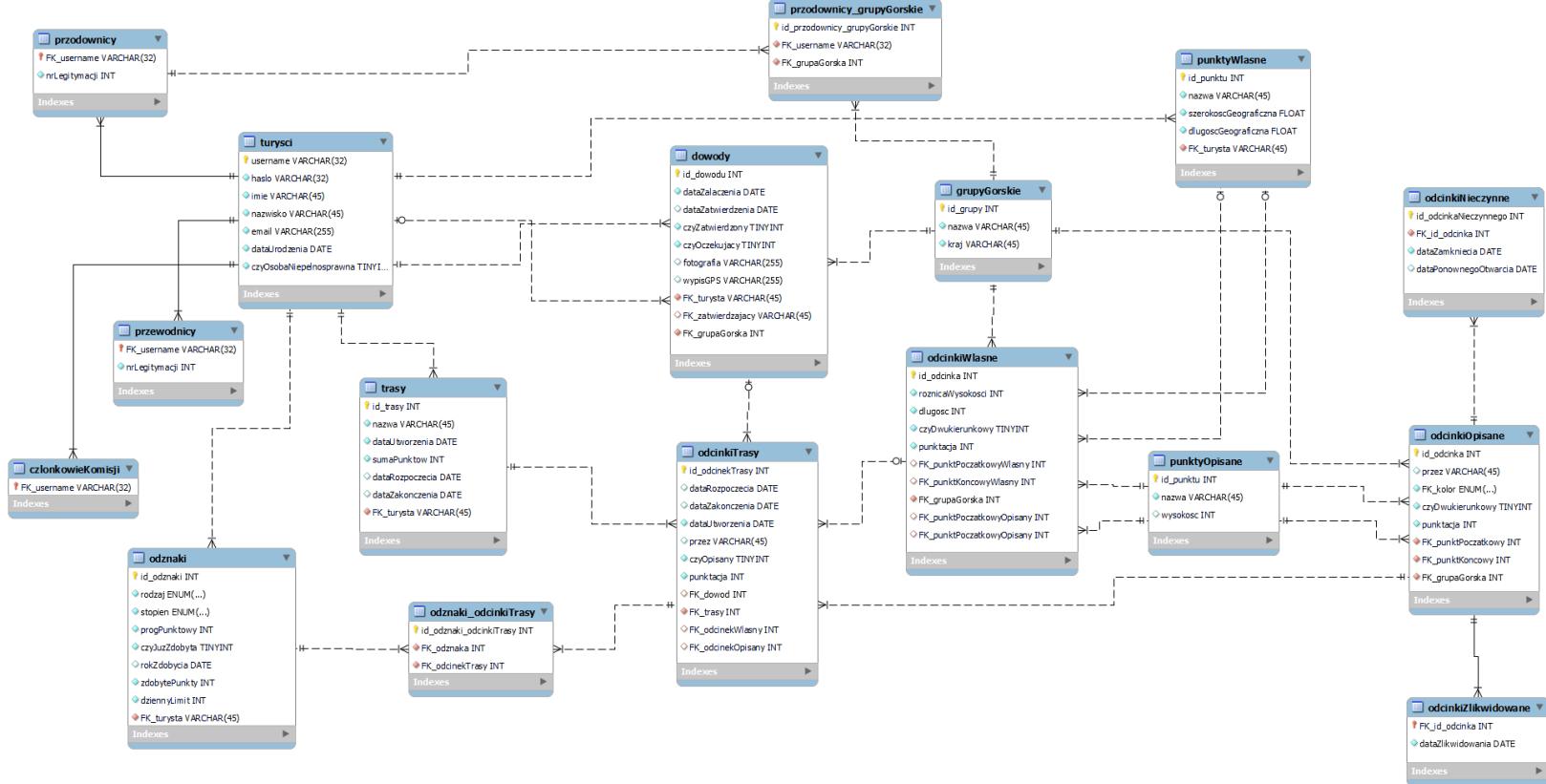
USUWANIE – Usuwanie punktów wyposażone jest w podpowiadanie nazw użytkownikowi, dlatego na samym początku wysyłane jest żądanie GET do backendu o zwrocenie wszystkich punktów własnych zalogowanego użytkownika. Każdy wpisywany znak w formularzu jest na bieżąco przetwarzany przez frontend. W momencie kliknięcia przycisku usunięcia, na frontendzie weryfikowane jest czy usuwany punkt występował we wcześniej pobranej liście punktów. Jeżeli tak, to wysyłany jest request DELETE do warstwy backendowej. Jeżeli nie, to wyświetlany jest stosowny komunikat dla użytkownika. Request DELETE jest odpowiednio przetwarzany i weryfikowany (czy punkt istnieje, czy należy do zalogowanego użytkownika, czy może być usunięty). Ponadto, standardowo weryfikowana jest stabilność połączenia z bazą danych.



EDYTOWANIE – Podobnie jak w przypadku usuwania lub szukania, na początku pobierana jest lista punktów własnych użytkownika (oraz weryfikowana stabilność połączenia). Wykorzystując wcześniej już nam znany „system podpowiadający nazw punktów”, oczekujemy i na bieżąco przetwarzamy wprowadzaną nazwę punktu. W momencie kliknięcia na przycisk potwierdzający edycję wybranego punktu, sprawdzamy czy jest on obecny we wcześniej pobranej liście i wysyłamy Request GET, podając dodatkowo id wybranego punktu. Request odpowiednio przetworzony zwraca nam wybrany punkt (tylko gdy połączenie z bazą jest stabilne). Użytkownikowi wyświetlany jest formularz do edycji (wypełniony danymi punktu). Formularz w czasie rzeczywistym jest przetwarzany poprzez frontend. Poprzez naciśnięcie przycisku edycji, weryfikowane jest czy pola obligatoryjne nie są puste (nazwa, szerokość i długość geograficzna). Jeżeli są niepuste, wysyłany jest Request PUT przekazujący nowe dane punktu. Dane te są odpowiednio weryfikowane (unikalność kluczowych atrybutów, czy istnieje, czy należy do zalogowanego użytkownika). Jeżeli weryfikacja przebiegnie pomyślnie – zwracane jest potwierdzenie edycji punktu (kod 200). W przeciwnym wypadku zwracany jest kod błędu 400 z odpowiednim komunikatem.

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

Projekt bazy danych



Relacyjna baza danych utworzona przy użyciu MySQL Workbench.

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

Etap IV

Podczas etapu IV uzupełnione zostały diagramy sekwencji dla przypadków użycia: Tworzenie trasy oraz Zgłaszanie prośby o zweryfikowanie trasy. Diagramy umieszczone w części dokumentacji etapu III.

Implementacja

1. Struktura kodu

```

    - backend
      - _pycache_
      - .venv
      - API
      - environment
      - models
      - serializers
      - services
      - templates
      - tests
      - app.py
    - frontend
      - cypress
      - node_modules
      - public
    - src
      - assets
      - components
        - Container
          - EvidenceConfirmation
          - LabeledPointManagement
          - OwnPointManagement
          - TourCreation
        - View
        - App.js
        - App.module.css
        - index.css
        - index.js
      - .gitignore
      - cypress.json
      - package-lock.json
      - package.json
      - README.md
      - .gitignore
      - README.md
  
```

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

Podczas implementacji powstały dodatkowe pakiety, nie uwzględnione wcześniej w diagramie pakietów: pakiet tests w pakiecie backend (na pliki testów jednostkowych), pakiet templates w pakiecie backend (na plik struktury dokumentacji kodu) oraz pakiet cypress w pakiecie frontend (na pliki z testami automatycznymi e2e). W strukturze znajdują się również pliki bibliotek (mają zaciemnione nazwy ze względu na to, że umieszczone są w .gitignore), pliki konfiguracyjne bibliotek (pliki z rozszerzeniem .json) a także pliki .gitignore oraz README.md.

Całość implementacji znajduje się pod adresem: <https://github.com/Vesperalin/Turysta-z-odznaka>

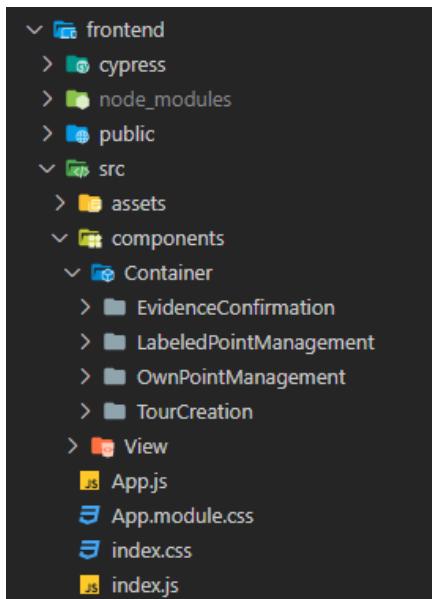
2. Generacja kodu

Kod wykorzystany w implementacji nie był generowany.

3. Wzorce projektowe

Wzorce projektowe używane w części frontendowej

Container/View – wzorzec posłużył do podzielenia komponentów na te odpowiedzialne za „stateful logic” i przechwytywanie danych z backendu (containers) oraz te odpowiedzialne za prezentowanie danych (views). W strukturze projektu można znaleźć wyraźny podział tych komponentów na pakiety: Container oraz View.



Użycie tego wzorca umożliwiło wydzielenie komponentów, które mogły być użyte w wielu miejscach aplikacji, a specyfikę swojego działania uzyskać poprzez właściwości z komponentów, w których zostały użyte. Sprawiło to, że część kodu jest wielorazowego użytku (w szczególności przyciski, linki, combobox oraz pola tekstowe). Ten zabieg zarówno zwiększa czytelność kodu, jak i zapobiega duplikacji kodu.

Przykład użycia:

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

```
import React from "react";

import styles from "./Button.module.css";

const Button = props => {
  return (
    <button onClick={props.onClick} className={styles.button}>
      <p>{props.text}</p>
    </button>
  );
};

export default Button;
```

Definicja komponentu Button (View)

```
import React from "react";
import { matchSorter } from 'match-sorter'

import ComboboxInputField from "../../View/ComboboxInputField/ComboboxInputField";
import Button from "../../View/Button/Button";
import styles from "./SearchForm.module.css";

const SearchForm = props => {
  const matchedPoints = nameMatch(props.term);

  function nameMatch(term) {
    return (
      term.trim() === ""
        ? null
        : matchSorter(props.labeledPoints, term, { keys: ['name'] })
    );
  }

  return [
    <div className={styles.formWrapper}>
      <h2>{props.title}</h2>
      <ComboboxInputField
        comboboxLabel="LabeledPoints"
        setTerm={props.setTerm}
        listElements={matchedPoints}
        inputPlaceholder={props.inputPlaceholder}
        noMatchInfo={props.noMatchInfo}
      />
      <Button
        text={props.buttonText}
        onClick={props.onSubmit}
      />
    </div>
  ];
};

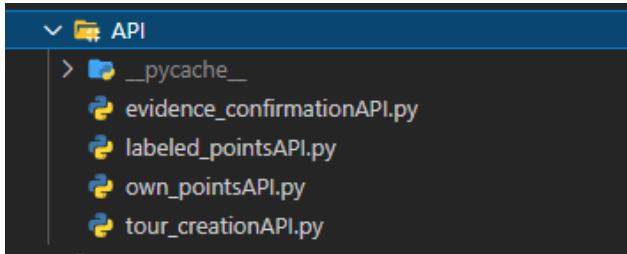
export default SearchForm;
```

Użycie komponentu Button w jednym z formularzy (Container). Przy okazji można zauważyc też, że komponent SearchForm wykorzystuje inny komponent służący do prezentacji: ComboboxInputField.

Wzorce projektowe używane w części backendowej

REST API – wzorzec posłużył do stworzenia aplikacyjnego interfejsu programistycznego, niezbędnego do integracji strony frontendowej z backendową. Interfejs API zgodny jest z zasadami projektowania REST. W strukturze kodu można zauważyc specjalnie wydzielony pakiet API, zawierający wszystkie interfejsy.

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>



Każdy taki interfejs API, przeznaczony do konkretnych przypadków użycia, zawiera zdefiniowane endpointy obsługujące żądania HTTP (GET, POST, PUT, DELETE). Wzorzec ten przede wszystkim ułatwia komunikację dwóch warstw aplikacji (warstwa logiki oraz warstwa prezentacji), ale także zwiększa czytelność kodu i izoluje warstwę logiki, przez co jest ona niezależna od warstwy prezentacji.

Przykład użycia:

```

46     @router.route('/<int:id>', methods=['GET'])
47     @auto.doc(groups=['own-point'])
48     def get_point(id):
49     """
50     ...
51     return get_own_point(id)
52
53
54     @router.route('', methods=['POST'])
55     @auto.doc(groups=['own-point'])
56     def add_point():
57     """
58     ...
59     return add_own_point()
60
61
62     @router.route('/<int:id>', methods=['PUT'])
63     @auto.doc(groups=['own-point'])
64     def update_point(id):
65     """
66     ...
67     return update_own_point(id)
68
69
70     @router.route('/<int:id>', methods=['DELETE'])
71     @auto.doc(groups=['own-point'])
72     def delete_point(id):
73     """
74     ...
75     return delete_own_point(id)

```

Zdefiniowane endpointy obsługujące żądania HTTP korzystają z metod dostarczonych przez pakiet services.

4. Dokumentacja kodu

Dokumentacja została wygenerowana dla REST API. Udokumentowanie tej części kodu jest bardzo ważne ze względu na to, że dzięki API spajamy część frontendową, z częścią backendową. Dokumentacja pozwala lepiej poznać wygląd, ciało oraz treść zapytań.

Do wygenerowania dokumentacji użyto biblioteki [flask-autodoc](#).

Dokumentacja jest dostępna na serwerze backendowym pod adresami:

- <http://127.0.0.1:5000/labeled-points/labele-point-API> - API punktów opisanych
- <http://127.0.0.1:5000/tour-creation/tour-creation-API> - API tworzenia trasy
- <http://127.0.0.1:5000/own-points/API> - API punktów własnych

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

- <http://127.0.0.1:5000/evidence-confirmation/API> - API zgłoszania odcinków trasy

Dokumentacja prezentuje się następujaco:

API tworzenia trasy

Documentation for tour creation API

Author: Klaudia Błażyczek

/tour-creation/check-name

Methods: POST, OPTIONS

Arguments: None

Description:

Returns information if user has a tour with given name.

Request body is a name of tour that is to be checked (in JSON format):

Example:

```
{
  "name": "Bieszczady 2022"
}
```

Returns data in format: result, status code:

- on success: True (in JSON format), 200

- when user has a tour with the name: {"message": "Trasa o wybranej nazwie już istnieje. Wybierz inną nazwę"}, 400

- when database error occurs: {"message": "Brak połączenia z bazą danych"}, 503

/tour-creation/labeled-segments

Methods: GET, HEAD, OPTIONS

Arguments: None

Description:

Returns all labeled segments.

Returns data in format: result, status code:

- on success: array of labeled segments (in JSON format), 200

- when database error occurs: {"message": "Brak połączenia z bazą danych"}, 503

/tour-creation/tour

Methods: POST, OPTIONS

Arguments: None

Description:

Adds new tour and its tour segments.

NOTE: doesn't check if segments contiguous in points - that should provided by endpoint user

Request body consist of: name of the tour, amount of GOT points and array of segments (in JSON format):

Example:

```
{
  "name": "Bieszczady 2022",
  "points": 8,
  "segments": [
    {
      "closed_segments": [],
      "color": "niebieski",
      "end_point_id": 43,
      "id": 10,
      "is_bidirectional": true,
      "liquidated_segment": null,
      "mountain_group_id": 1,
      "points": 7,
      "start_point_id": 12,
      "through": null
    },
    {
      "closed_segments": [],
      "color": "niebieski",
      "end_point_id": 44,
      "id": 13,
      "is_bidirectional": true,
      "liquidated_segment": null,
      "mountain_group_id": 1,
      "points": 1,
      "start_point_id": 43,
      "through": null
    }
  ]
}
```

Returns data in format: result, status code:

- on success: array of created tour segments (in JSON format), 200

- when database error occurs: {"message": "Brak połączenia z bazą danych"}, 503

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

API punktów opisanych

Documentation for labeled points API

Author: Klaudia Błażyczek

/labeled-points

Methods: [GET](#), HEAD, OPTIONS

Arguments: *None*

Description:

Returns all labeled points.

Returns data in format: result, status code:

- on success: array of labeled points (in JSON format), 200
- when database error occurs: {"message": "Brak połączenia z bazą danych"}, 503

/labeled-points

Methods: POST, OPTIONS

Arguments: *None*

Description:

Adds new labeled point.

Request body is a labeled point (in JSON format):

Example:

```
{
  "name": "Bukowe Berdo",
  "height": null
}
```

Returns data in format: result, status code:

- on success: {"message": "Punkt został pomyślnie dodany"}, 200
- when height is not correct: {"message": "Niepoprawna wartość wysokości"}, 400
- when name is not unique: {"message": "Punkt o takiej nazwie już istnieje. Wybierz inną nazwę"}, 400
- when database error occurs: {"message": "Brak połączenia z bazą danych"}, 503

/labeled-points/<int:id>

Methods: GET, HEAD, OPTIONS

Arguments: *id*

Description:

Returns labeled point with specified id.

Returns data in format: result, status code:

- on success: labeled point (in JSON format), 200
- when point with id not exists: {"message": "Punkt o wybranej nazwie nie istnieje"}, 400
- when database error occurs: {"message": "Brak połączenia z bazą danych"}, 503

/labeled-points/<int:id>

Methods: OPTIONS, PUT

Arguments: *id*

Description:

Updates labeled point with specified id.

Request body is a labeled point (in JSON format):

Example:

```
{
  "name": "Bukowe Berdo",
  "height": null
}
```

Returns data in format: result, status code:

- on success: {"message": "Punkt został pomyślnie edytowany"}, 200
- when point with id not exists: {"message": "Punkt o wybranej nazwie nie istnieje"}, 400
- when point is used in segments and can't be edited: {"message": "Punkt jest już używany w odcinkach. Nie można go edytować"}, 400
- when point with specified name already exists: {"message": "Punkt o takiej nazwie już istnieje. Wybierz inną nazwę"}, 400
- when height is not correct: {"message": "Niepoprawna wartość wysokości"}, 400
- when database error occurs: {"message": "Brak połączenia z bazą danych"}, 503

/labeled-points/<int:id>

Methods: DELETE, OPTIONS

Arguments: *id*

Description:

Deletes labeled point with specified id.

Returns data in format: result, status code:

- on success: {"message": "Punkt został pomyślnie usunięty"}, 200
- when point is used in segments and can't be removed: {"message": "Punkt jest już używany w odcinkach. Nie można go usuwać"}, 400
- when point with id not exists: {"message": "Punkt o wybranej nazwie nie istnieje"}, 400
- when database error occurs: {"message": "Brak połączenia z bazą danych"}, 503

/labeled-points/like

Methods: [GET](#), HEAD, OPTIONS

Arguments: *None*

Description:

Returns all labeled points. Catches requests: /like/<string:like> where argument is empty.

Returns data in format: result, status code:

- on success: array of labeled points (in JSON format), 200
- when database error occurs: {"message": "Brak połączenia z bazą danych"}, 503

/labeled-points/like/<string:like>

Methods: GET, HEAD, OPTIONS

Arguments: *like*

Description:

Returns labeled points that contains given phrase.

Returns data in format: result, status code:

- on success: array of labeled points (in JSON format), 200
- when database error occurs: {"message": "Brak połączenia z bazą danych"}, 503

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

API zgłoszenia odcinków trasy

Documentation for evidence confirmation API

Author: Justyna Małuszyńska

/evidence-confirmation/evidence

Methods: OPTIONS, POST

Arguments: None

Description:

Adds new evidences to selected tour segments. Updates tour segments with start and end dates.

Request body consist of:

Example:

```
{
  "attachments": [
    {"value": "dowod.png", "mountainGroup": 2, "tourSegments": [
      {"id": 27, "startDate": "2020-01-01", "endDate": "2020-01-03"},
      {"id": 28, "startDate": "2020-01-05", "endDate": "2020-01-06"}
    ]}
  ],
  "verifying": [
    {"idVerifying": 123456, "tourSegments": [
      {"id": 29, "startDate": "2020-01-04", "endDate": "2020-01-04"},
      {"id": 32, "startDate": "2020-01-05", "endDate": "2020-01-05"}
    ]}
  ]
}
```

Returns data in format: result, status code:

- on success: array of reported tour segments (in JSON format) with evidences, 200
- when database error occurs: {"message": "Brak połączenia z bazą danych"}, 503

/evidence-confirmation/guide/<int:id_guide>

Methods: OPTIONS, GET, HEAD

Arguments: id_guide

Description:

Checks if a guide with given id number exists.

Returns data in format: result, status code:

- on success: {"message": "OK"}, 200
- when guide with specified id number not exist: {"message": "Przewodnik o podanym numerze legitymacji nie istnieje"}, 404
- when database error occurs: {"message": "Brak połączenia z bazą danych"}, 503

/evidence-confirmation/segments/<int:tour_id>

Methods: OPTIONS, GET, HEAD

Arguments: tour_id

Description:

Returns all unreported tour segments with specified tour id.

Returns data in format: result, status code:

- on success: array of unreported tour segments (in JSON format), 200
- when database error occurs: {"message": "Brak połączenia z bazą danych"}, 503

/evidence-confirmation/tours

Methods: OPTIONS, [GET](#), HEAD

Arguments: None

Description:

Returns all user tours.

Returns data in format: result, status code:

- on success: array of user tours (in JSON format), 200
- when database error occurs: {"message": "Brak połączenia z bazą danych"}, 503

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

API odcinków własnych

Documentation for own points API

Author: Justyna Małuszyńska

/own-points

Methods: OPTIONS, [GET](#), HEAD

Arguments: None

Description:

- Returns all user own points.
- Returns data in format: result, status code:
 - on success: array of own points (in JSON format), 200
 - when database error occurs: {"message": "Brak połączenia z bazą danych"}, 503

/own-points

Methods: OPTIONS, POST

Arguments: None

Description:

- Adds new own point.
- Request body is an own point (in JSON format):

Example:

```
{
  "name": "Klimczok Skala",
  "longitude": 49.357236,
  "latitude": 26.832768
}
```

Returns data in format: result, status code:

- on success: {"message": "Punkt został pomyślnie dodany"}, 200
- when latitude is not correct: {"message": "Niepoprawne dane. Szerokość geograficzna musi być liczbą rzeczywistą"}, 400
- when longitude is not correct: {"message": "Niepoprawne dane. Długość geograficzna musi być liczbą rzeczywistą"}, 400
- when name is not unique: {"message": "Punkt o takiej nazwie już istnieje. Wybierz inną nazwę"}, 400
- when coordinates are not unique: {"message": "Punkt z podanymi współrzędnymi geograficznymi już istnieje"}, 400
- when database error occurs: {"message": "Brak połączenia z bazą danych"}, 503

/own-points/<int:id>

Methods: OPTIONS, GET, HEAD

Arguments: id

Description:

- Returns user own point with specified id.
- Returns data in format: result, status code:
 - on success: own point (in JSON format), 200
 - when requested point do not belongs to user: {"message": "Punkt nie należy do tego użytkownika"}, 400
 - when point with id not exists: {"message": "Punkt o wybranej nazwie nie istnieje"}, 400
 - when database error occurs: {"message": "Brak połączenia z bazą danych"}, 503

/own-points/<int:id>

Methods: OPTIONS, PUT

Arguments: id

Description:

- Updates own point with specified id.
- Request body is an own point (in JSON format):
- Example:

```
{
  "name": "Klimczok Skala",
  "longitude": 49.357236,
  "latitude": 26.832768
}
```

- Returns data in format: result, status code:
 - on success: {"message": "Punkt został pomyślnie edytowany"}, 200
 - when point with id not exists: {"message": "Punkt o wybranej nazwie nie istnieje"}, 400
 - when requested point do not belongs to user: {"message": "Punkt nie należy do tego użytkownika"}, 400
 - when point is used in segments and can't be edited: {"message": "Punkt jest już używany w odcinkach. Nie można go edytować"}, 400
 - when point with specified name already exists: {"message": "Punkt o takiej nazwie już istnieje. Wybierz inną nazwę"}, 400
 - when point with specified coordinates already exists: {"message": "Punkt z podanymi współrzędnymi geograficznymi już istnieje"}, 400
 - when latitude is not correct: {"message": "Niepoprawne dane. Szerokość geograficzna musi być liczbą rzeczywistą"}, 400
 - when longitude is not correct: {"message": "Niepoprawne dane. Długość geograficzna musi być liczbą rzeczywistą"}, 400
 - when database error occurs: {"message": "Brak połączenia z bazą danych"}, 503

/own-points/<int:id>

Methods: OPTIONS, DELETE

Arguments: id

Description:

- Deletes own point with specified id.
- Returns data in format: result, status code:
 - on success: {"message": "Punkt został pomyślnie usunięty"}, 200
 - when point is used in segments and can't be removed: {"message": "Punkt jest już używany w odcinkach. Nie można go usuwać"}, 400
 - when point with id not exists: {"message": "Punkt o wybranej nazwie nie istnieje"}, 400
 - when requested point do not belongs to user: {"message": "Punkt nie należy do tego użytkownika"}, 400
 - when database error occurs: {"message": "Brak połączenia z bazą danych"}, 503

/own-points/like/

Methods: OPTIONS, [GET](#), HEAD

Arguments: None

Description:

- Returns all user own points. Catches requests: own-points/like/<string:like> where argument "like" is empty.
- Returns data in format: result, status code:
 - on success: array of own points (in JSON format), 200
 - when database error occurs: {"message": "Brak połączenia z bazą danych"}, 503

/own-points/like/<string:like>

Methods: OPTIONS, GET, HEAD

Arguments: like

Description:

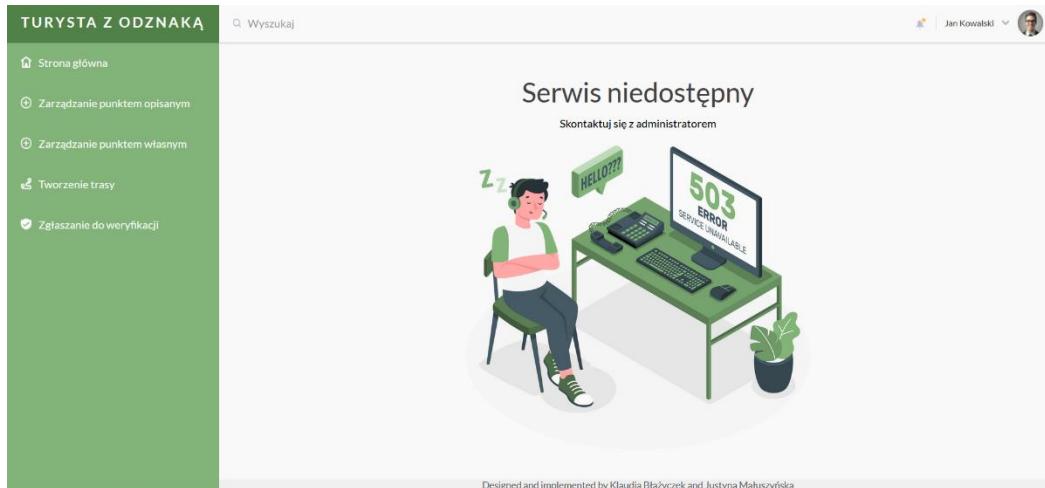
- Returns user own points that contains given phrase.
- Returns data in format: result, status code:
 - on success: array of own points (in JSON format), 200
 - when database error occurs: {"message": "Brak połączenia z bazą danych"}, 503

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

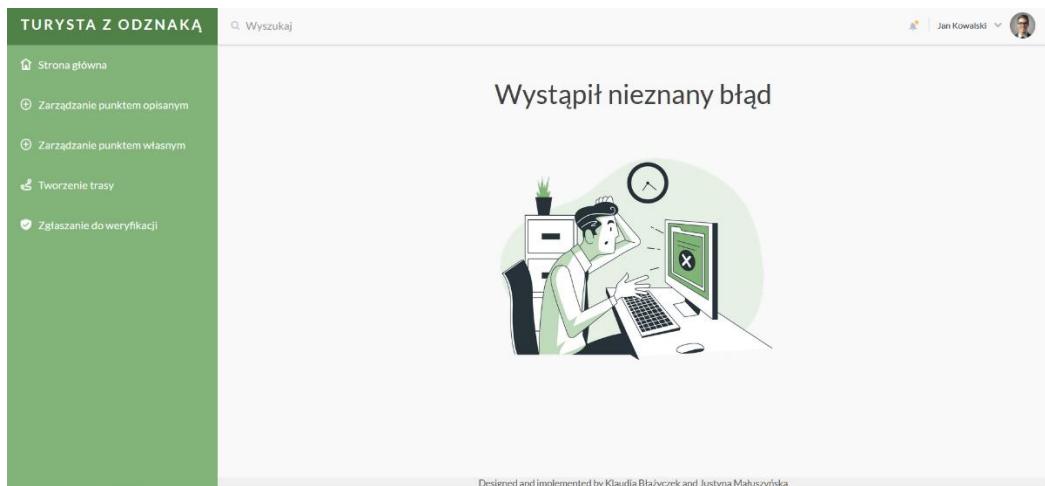
5. Mockupy, a realizacja – dokumentacja rozbieżności

Ogólne

- Podczas implementacji dodano stronę informującą o błędzie w połączeniu z serwerem. W mockupach taka strona nie istniała



- Podczas implementacji dodano stronę informującą o nieznanym błędzie. W mockupach taka strona nie istniała



- Dodatkowo podczas implementacji dodano stopkę z autorami projektu oraz linkami do profilów na GitHubie (stopka została wprowadzona na wszystkich stronach aplikacji webowej)

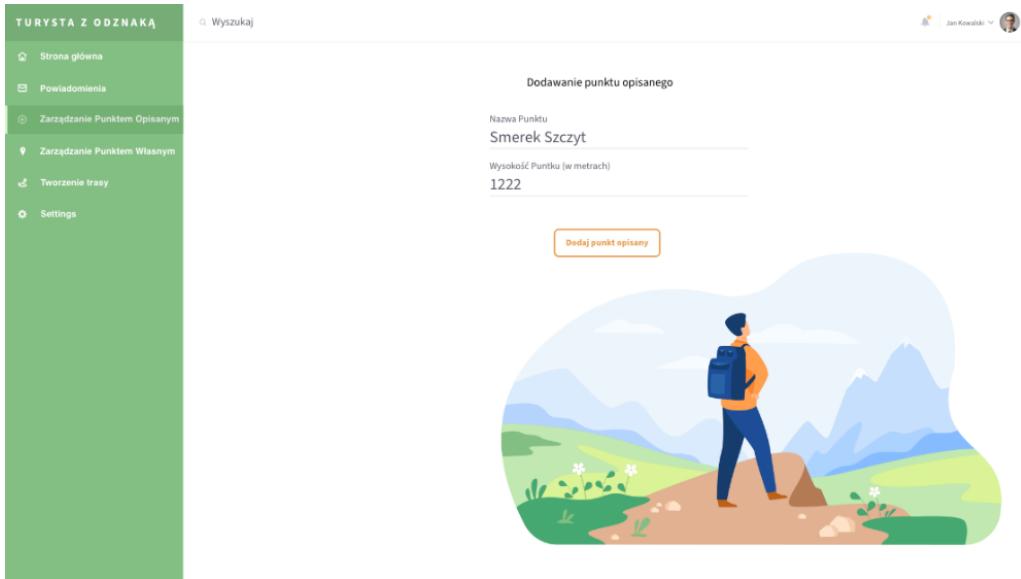
Przypadek użycia: Zarządzanie punktem opisanym (Klaudia Błażyczek)

- Na mockupie w przypadku pól tekstowych oraz comboboxów, każde z nich było podpisane tekstem mówiącym, co należy wpisać w dane pole. Podczas implementacji zrezygnowano z tego pomysłu. Zamiast tego wykorzystano placeholdery.

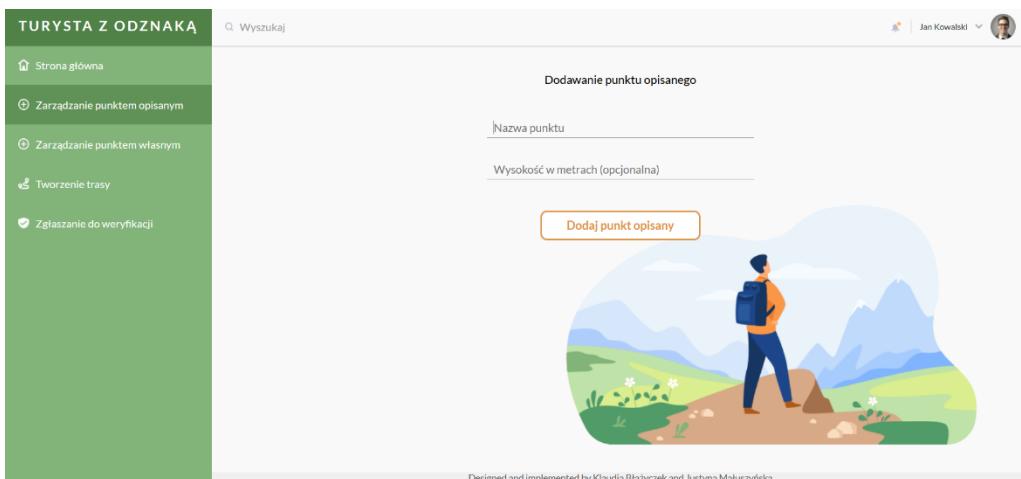
“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

Przykład:

Mockup



Implementacja



“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

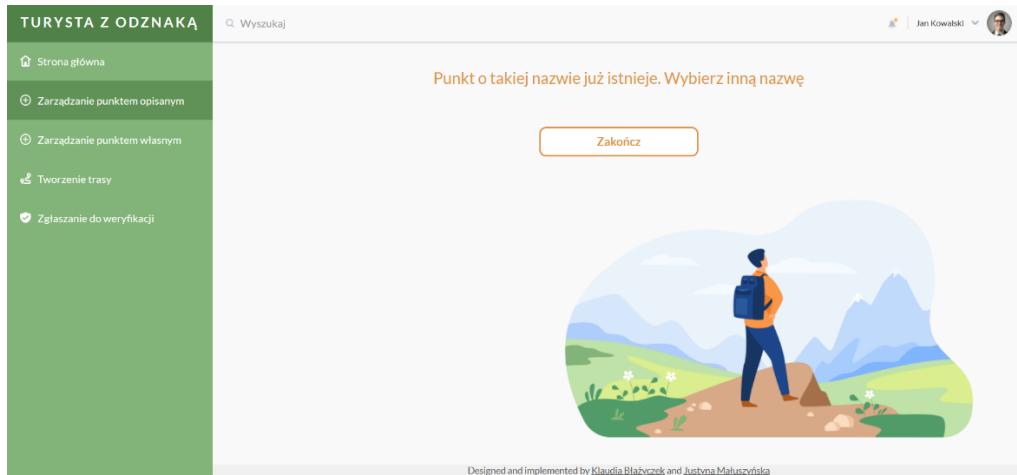
2. Na mockupach nie został uwzględniony przypadek, w którym podczas dodawania punktu opisanego ktoś nie wpisuje nazwy punktu. Podeczas implementacji dodano wyświetlanie tego błędu użytkownikowi i umożliwienie wprowadzenia nazwy.

3. W przypadku, gdy podczas dodawania punktu, program zweryfkuje, że punkt opisany o takiej nazwie już istnieje, komunikat o tym jest przekazywany w inny sposób – zrezygnowano z czerwonych napisów.

Mockup

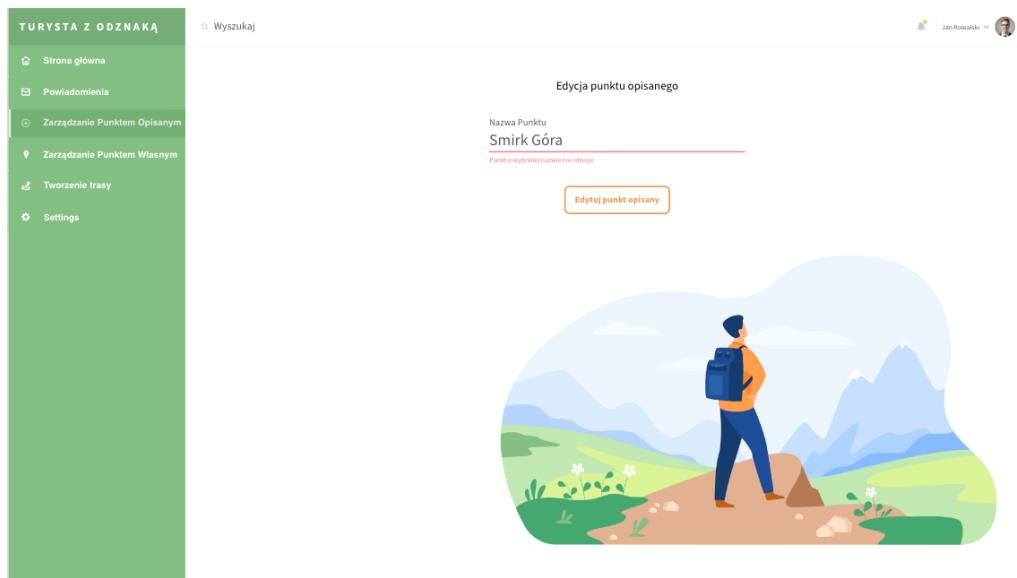
Implementacja

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>



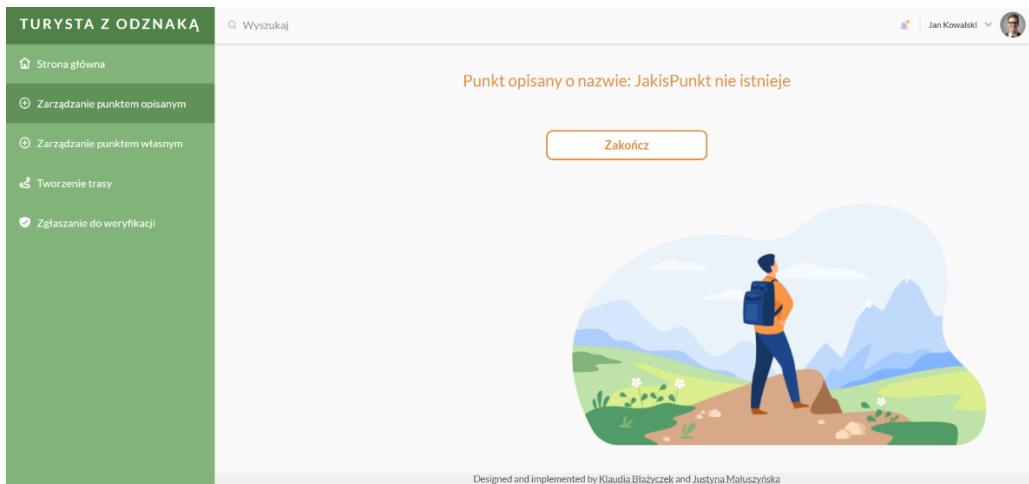
4. Podczas edycji punktu, gdy w formularzu wyszukującym użytkownik wpisze nazwę nieistniejącego punktu, błąd ten zostanie inaczej pokazany użytkownikowi

Mockup



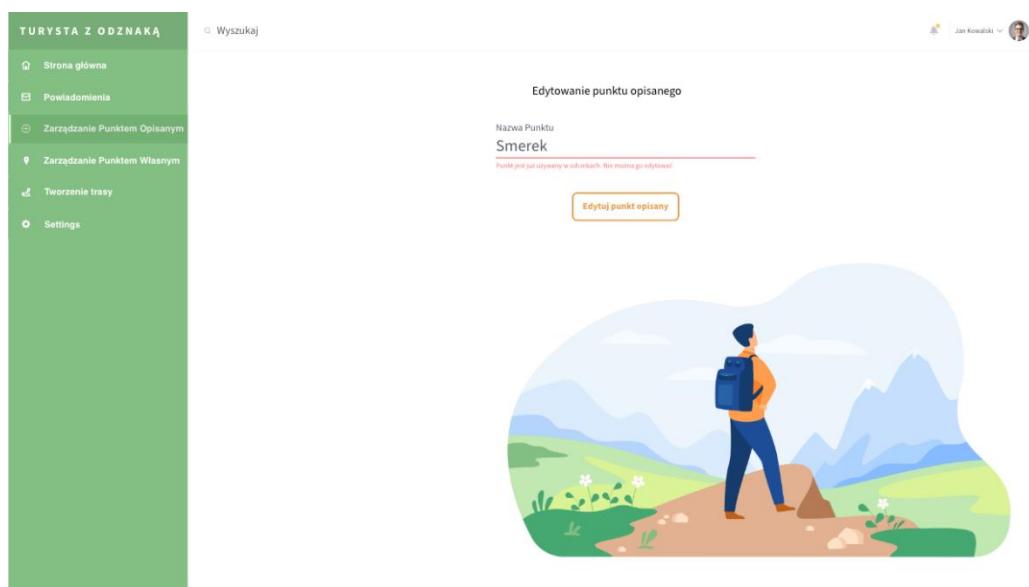
Implementacja

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>



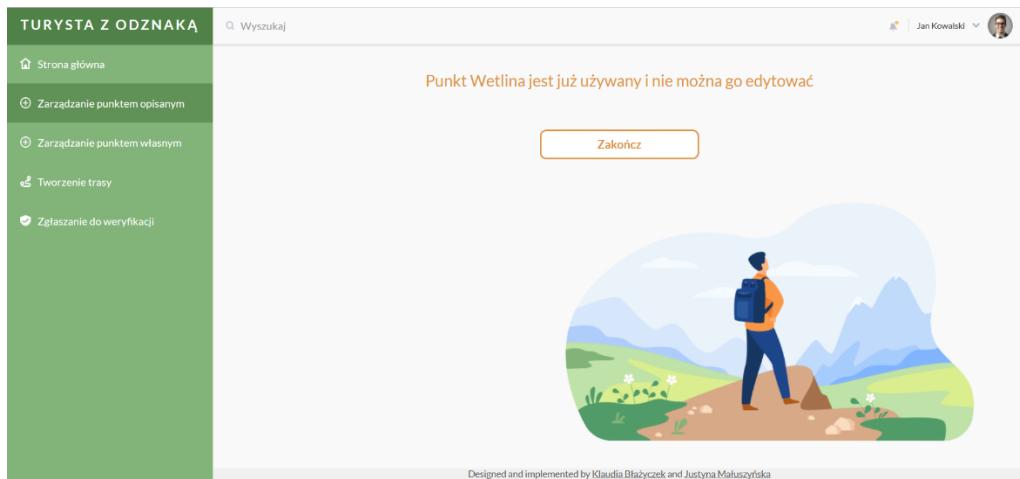
5. Podczas edycji punktu, gdy w formularzu wyszukującym użytkownik wpisze nazwę punktu, który nie może być edytowany ze względu na to, że jest już używany w odcinkach, błąd ten zostanie inaczej pokazany użytkownikowi

Mockup



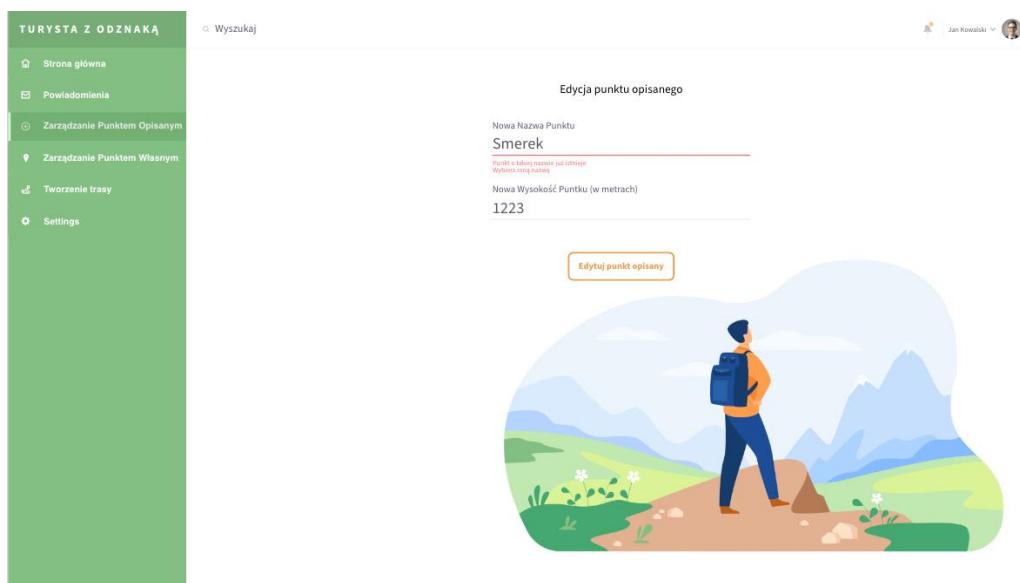
Implementacja

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>



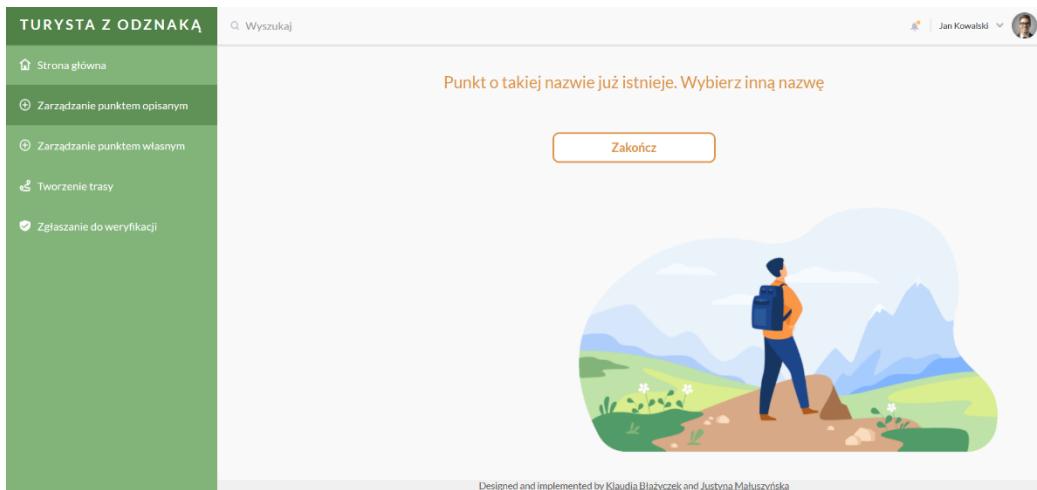
6. Podczas edycji punktu, gdy w formularzu edycji punktu użytkownik wpisze nazwę punktu, który istnieje, błąd ten zostanie inaczej pokazany użytkownikowi

Mockup



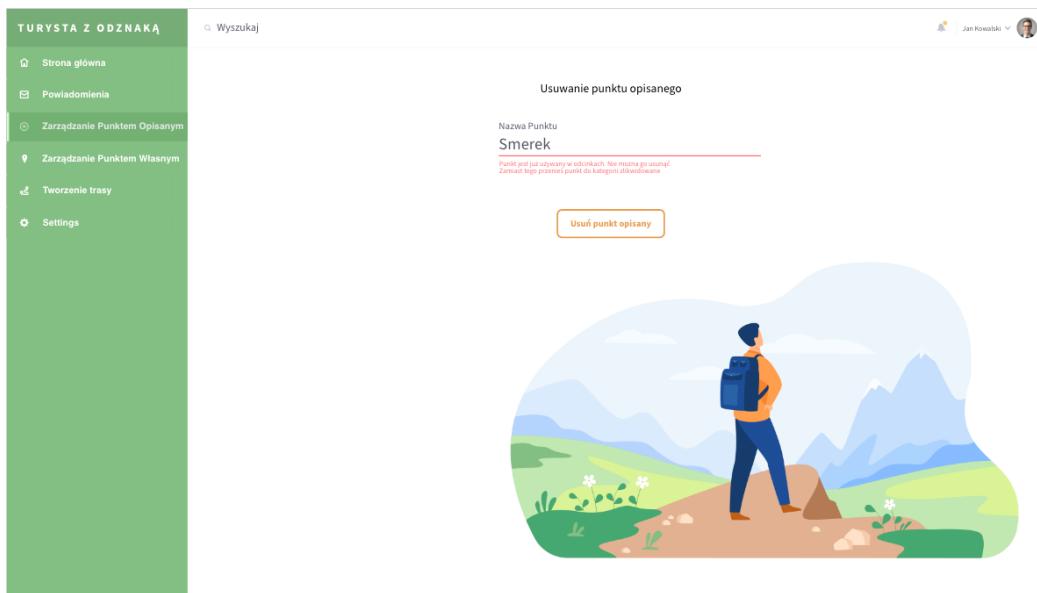
Implementacja

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>



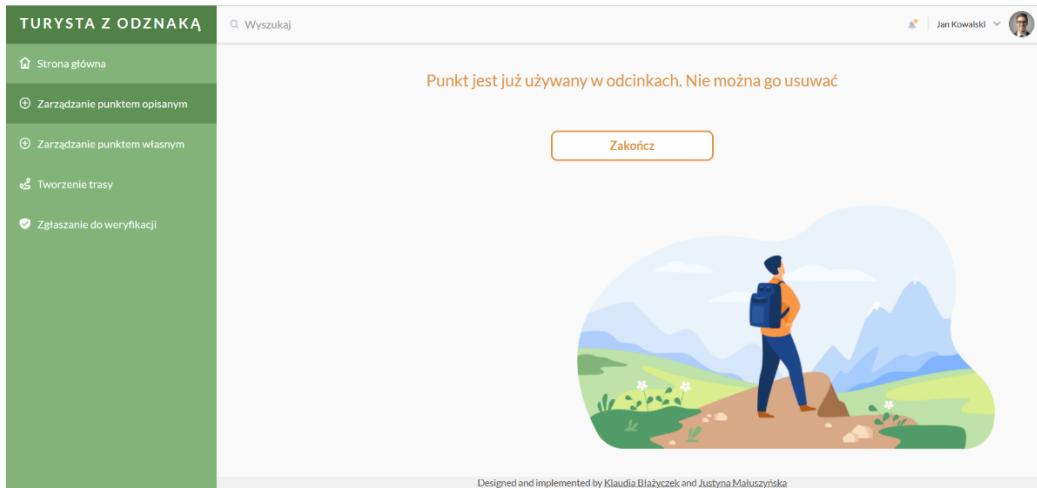
7. Podczas usuwania punktu, gdy w formularzu wyszukującym użytkownik wpisze nazwę punktu, który nie może być usunięty ze względu na to, że jest już używany w odcinkach, błąd ten zostanie inaczej pokazany użytkownikowi

Mockup



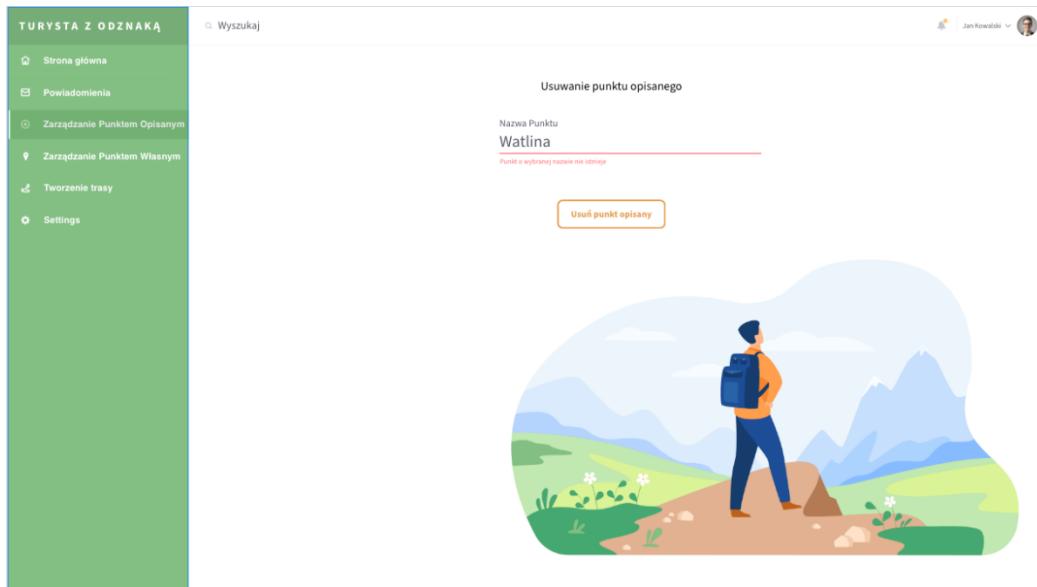
Implementacja

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>



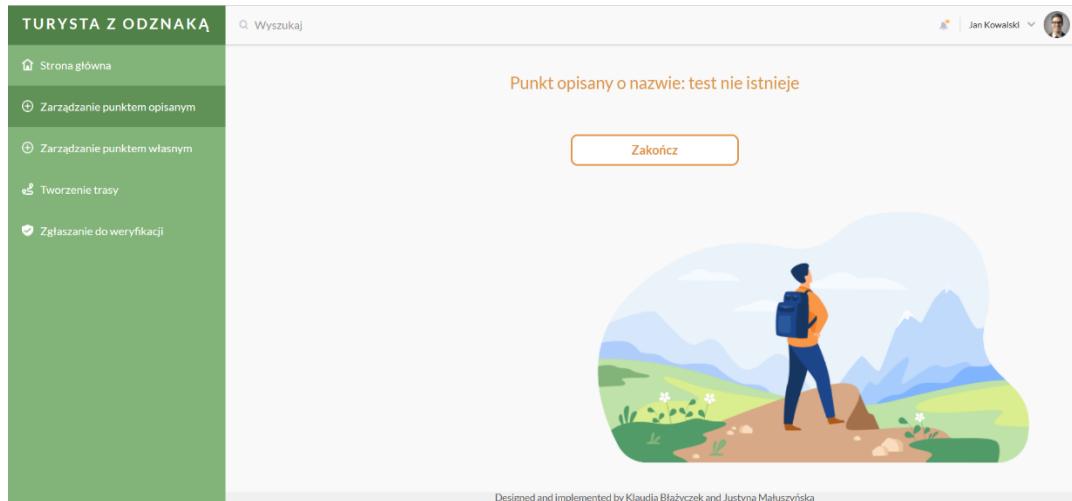
- 8. Podczas usuwania punktu, gdy w formularzu wyszukującym użytkownik wpisze nazwę nieistniejącego punktu, błąd ten zostanie inaczej pokazany użytkownikowi**

Mockup



Implementacja

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>



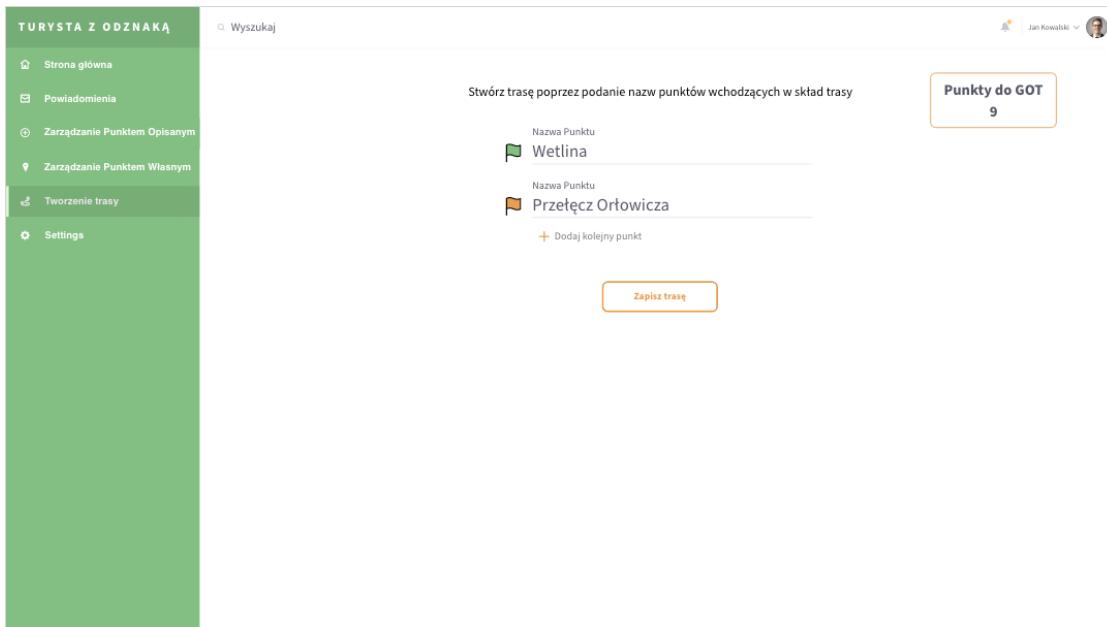
Podczas implementacji nastąpiły też drobne zmiany w komunikatach błędów. Ich treść nie ma jednak większego wpływu na wygląd interfejsu.

Jeżeli podczas wyszukiwania punktów w formularzu zatwierdzi się puste pole, zostaną wyświetlane wszystkie punkty opisane.

Przypadek użycia: Tworzenie trasy (Klaudia Błażyczek)

Podczas implementacji powstał pomysł, który bardzo zmienił wygląd tego przypadku użycia. Wpłynęło to też po części na sposób działania przypadku użycia (diagram aktywności nie jest już tak adekwatny – zostało to opisane przy diagramie sekwencji). Zmiana ta jest jak najbardziej pozytywna. Dzięki niej użytkownik może popełnić o wiele mniej błędów, a co za tym idzie, chętniej będzie korzystał z funkcjonalności aplikacji webowej. Zmiana to też ukłon w stronę mniej doświadczonych turystów, ponieważ o wiele łatwiej będzie im ułożyć spójne trasy swoich wycieczek.

Pierwotnie, formularz do wybierania punktów tworzących odcinki miał wyglądać następująco



“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

Podczas implementacji, został on podzielony na dwa osobne formularze

Designed and implemented by Klaudia Blażycka and Justyna Małuszewska

Pierwszy formularz służy do wprowadzania punktu początkowego trasy. Pole do wpisywania nazwy jest comboboxem, co za tym idzie, podpowiada użytkownikowi pasujące opcje.

Designed and implemented by Klaudia Blażycka and Justyna Małuszewska

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

W przypadku, gdy użytkownik zatwierdzi punkt, który nie istnieje pojawi się następujący komunikat

The screenshot shows a user interface for managing routes. On the left, a sidebar menu includes: Strona główna, Zarządzanie punktem opisanym, Zarządzanie punktem własnym, Tworzenie trasy, and Zgłaszczenie do weryfikacji. The main area has a search bar at the top right. A button labeled "Stwórz trasę poprzez podanie punktów tworzących odcinki" is visible. Below it, there is an error message: "posdhsgdhsgh" followed by "Punkt opisany o nazwie: posdhsgdhsgh nie istnieje". A button labeled "Dodaj punkt startowy" is present. In the top right corner, there is a user profile for "Jan Kowalski" and a box for "Punkty do GOT" with a value of 0.

Kiedy podana nazwa punktu początkowego jest poprawna, użytkownik zostanie przeniesiony do kolejnego formularza, który umożliwia wybranie kolejnych punktów trasy.

This screenshot shows the next step in route creation after selecting a starting point. The sidebar and top navigation are identical to the previous screenshot. The main area now displays a placeholder point "Polanki" with a location pin icon. Below it is a dropdown menu labeled "Wybierz kolejny punkt z listy" containing a single item. At the bottom of the form are two buttons: "+ Dodaj punkt" and "- Usuń ostatni punkt". A large orange "Zatwierdź trasę" button is centered at the bottom. The bottom right corner shows the user profile for "Jan Kowalski" and the "Punkty do GOT" box with a value of 0.

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

Designed and implemented by Klaudia Błażyczek and Justyna Małuszynska

Z listy rozwijanej turysta może wybierać kolejne punkty trasy tworzące spójne odcinki. Z listy tej wykluczane są punkty, które z poprzednim tworzą odcinek zamknięty lub nieczynny. Przycisk „Dodaj punkt” doda do trasy wybrany z listy punkt, natomiast przycisk „Usuń ostatni punkt” usunie ostatni z wybranych punktów trasy. W przypadku, gdy usunięty zostanie punkt początkowy, turysta zostanie cofnięty do poprzedniego formularza.

Jeżeli turysta będzie chciał dodać pusty punkt, zostanie wyświetlony następujący komunikat

Designed and implemented by Klaudia Błażyczek and Justyna Małuszynska

Po zatwierdzeniu wybranych punktów, turysta zostanie poproszony o wybranie nazwy trasy. W przypadku podawania nazwy istniejącej trasy turysty, ta część jest bardzo podobna do tej zaprezentowanej w mockupach. Różni się tylko nieco sposób prezentacji komunikatu

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

Mockup

Implementacja

W przypadku, gdy nazwa jest unikatowa dla trasy turysty, po zapisaniu trasy i odcinków trasy do bazy danych, zostanie zaprezentowane podsumowanie utworzonej trasy.

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

The screenshot shows a user interface for managing routes. On the left, a sidebar menu lists options: Strona główna, Zarządzanie punktem opisanym, Zarządzanie punktem własnym, Tworzenie trasy, and Zgłoszanie do weryfikacji. The main area displays a summary of a recorded route: "Zapisano trasę: Bieszczady 2022" and "Liczba punktów do GOT: 14". A table shows two segments of the route: 1. Polanki - Łopiankę (Konięc, Bieszczady, 13 points) and 2. Durna - Berdo (Bieszczady, 1 point). Below the table is a "Zakończ" button and a stylized illustration of mountains and a sun. At the bottom, it says "Designed and implemented by Klaudia Błażyczek and Justyna Maluszynska".

Dzięki tak zaprojektowanemu interfejsowi, system uniemożliwia turystie pomyłkę w nazwie punktów, dodanie punktów, które nie tworzą odcinka z poprzednim lub są zlikwidowane/czasowo zamknięte. Jedyne błędy jakie mogą się więc pojawić to: podanie złego punktu początkowego i podanie istniejącej nazwy trasy (oczywiście mogą też pojawić się błędy z serwerami, ale dotyczy to każdego przypadku użycia).

Przypadek użycia: Zarządzanie punktem własnym (Justyna Maluszyńska)

Realizacja przypadku użycia nie odbiega znacząco od wcześniej prezentowanych mockupów. Podstawowe różnice pomiędzy tymi dwoma interfejsami:

1. Zrezygnowano z wcześniej proponowanych na mockupach podpisów konkretnych pól w formularzach (tzw. labeli) na rzecz podpowiedzi znajdujących się już w konkretnym polu formularza (placeholder):

Mockup:

The screenshot shows a form for adding a new point. The sidebar menu includes: Strona główna, Powiadomienia, Zarządzanie Punktem Opisanym, Zarządzanie Punktem Własnym, Tworzenie trasy, and Settings. The main area is titled "Dodawanie punktu własnego" and contains fields for "Nazwa punktu" (with placeholder "Nazwa punktu"), "Szerokość geograficzna" (with placeholder "Szerokość geograficzna"), and "Długość geograficzna" (with placeholder "Długość geograficzna"). A "Dodaj punkt własny" button is at the bottom. Below the form is a stylized illustration of two hikers on a path.

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

Realizacja:

- Zmieniono sposób informowania użytkownika o błędzie. Zrezygnowano z podpowiedzi o błędzie w kolorze czerwonym na rzecz dużych komunikatów w kolorze pomarańczowym, kończących wykonywaną akcję w następujących przypadkach:

Mockup: (zła nazwa podczas edycji punktu)

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

Realizacja: (zła nazwa podczas edycji punktu)

The screenshot shows the application's navigation bar with 'TURYSTA Z ODZNAKĄ' and a search bar. The main content area displays an error message: 'Punkt o takiej nazwie już istnieje. Wybierz inną nazwę' (Point with such a name already exists. Choose another name). Below the message is a large orange button labeled 'Zakończ' (Finish). To the right is a stylized illustration of a hiker standing on a path in a mountainous landscape. At the bottom of the screen, a small text indicates: 'Designed and Implemented by Klaudia Brzyzceek and Justyna Matuszyńska'.

Mockup: (usuwanie nieistniejącego punktu)

The screenshot shows the application's navigation bar with 'TURYSTA Z ODZNAKĄ' and a search bar. The main content area displays a message: 'Usuwanie punktu własnego' (Deleting own point) above a text input field containing 'Schronisko w Kamieńcu'. Below the input field is a red error message: 'Punkt o wybranym nazwie nie istnieje' (Point with selected name does not exist). A blue button labeled 'Usuń punkt własny' (Delete own point) is centered below the input field. To the right is a stylized illustration of two hikers walking along a winding path in a green landscape. At the bottom of the screen, a small text indicates: 'Designed and Implemented by Klaudia Brzyzceek and Justyna Matuszyńska'.

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

Realizacja: (usuwanie nieistniejącego punktu)

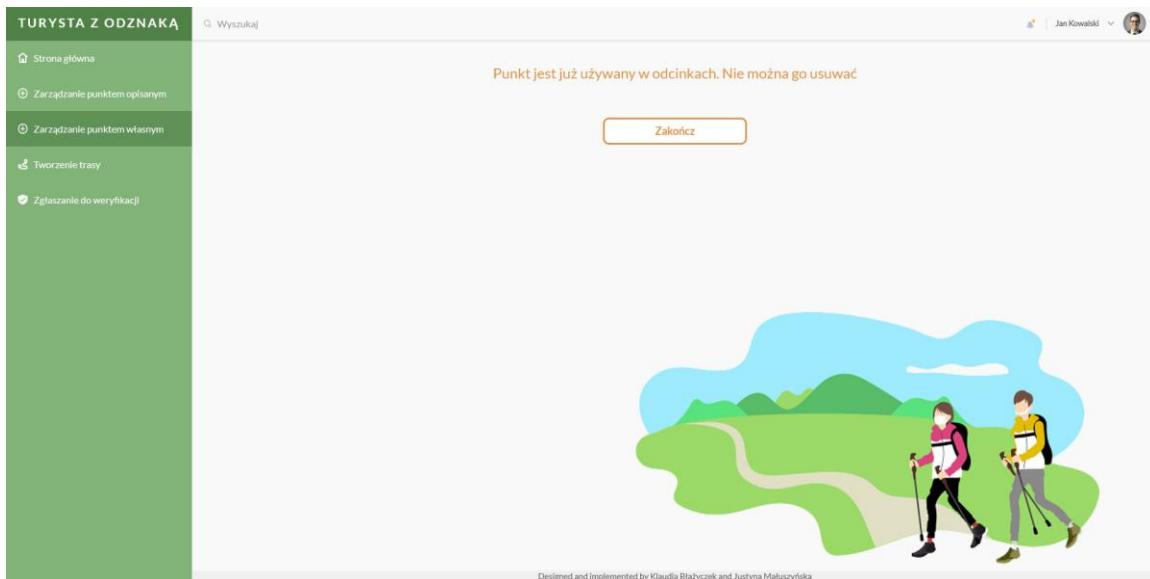
The screenshot shows a user interface for managing tourism points. On the left, a sidebar titled "TURYSTA Z ODZNAKĄ" lists navigation options: Strona główna, Zarządzanie punktem opisanym, Zarządzanie punktem własnym, Tworzenie trasy, and Zgłaszać do weryfikacji. The main area has a search bar labeled "Wyszukaj" and a message "Punkt własny o nazwie: usun nie istnieje" above a button labeled "Zakoncz". Below the button is a stylized illustration of two hikers on a path. At the bottom, a footer note reads "Designed and implemented by Klaudia Błażyczek and Justyna Małuszynska".

Mockup: (usuwanie punktu używanego w odcinkach)

The mockup shows a similar user interface. The sidebar includes "Strona główna", "Powiadomienia", "Zarządzanie Punktem Opisanym", "Zarządzanie Punktem Własnym", "Tworzenie trasy", and "Settings". The main area displays a message "Usuwanie punktu opisanego" above a "Nazwa Punktu" field containing "Schronisko przy Chełmiance". A note below states "Punkt jest już używany w odcinkach. Nie można go usunąć." and features a button labeled "Usuń punkt własny". A stylized illustration of two hikers is present at the bottom.

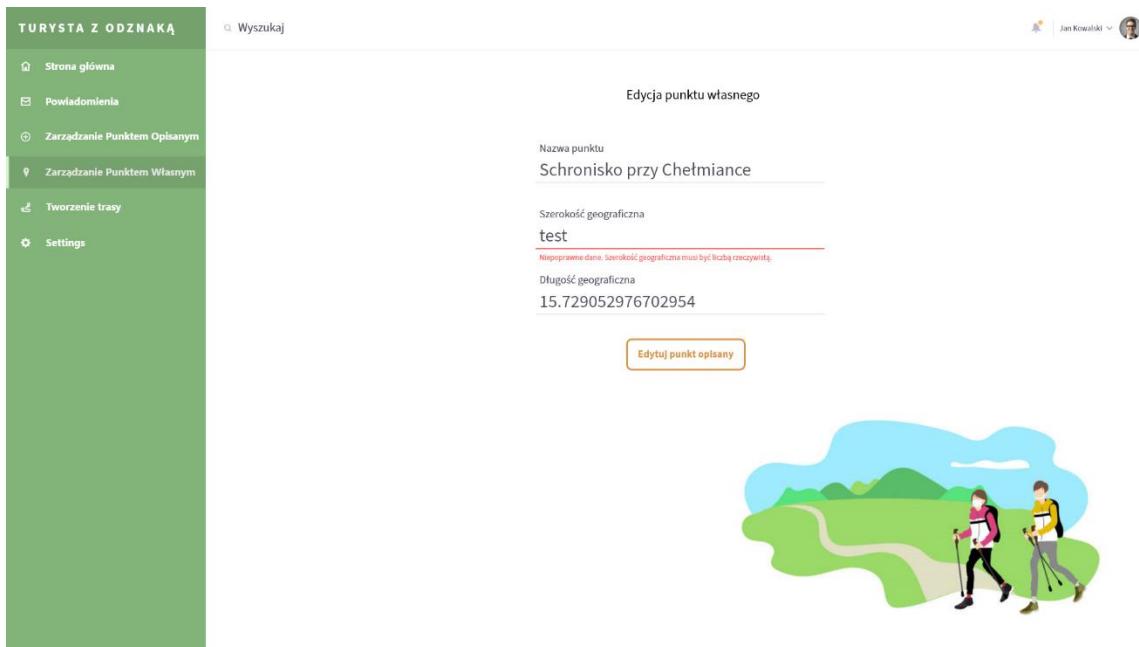
“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

Realizacja: (usuwanie punktu używanego w odcinkach)



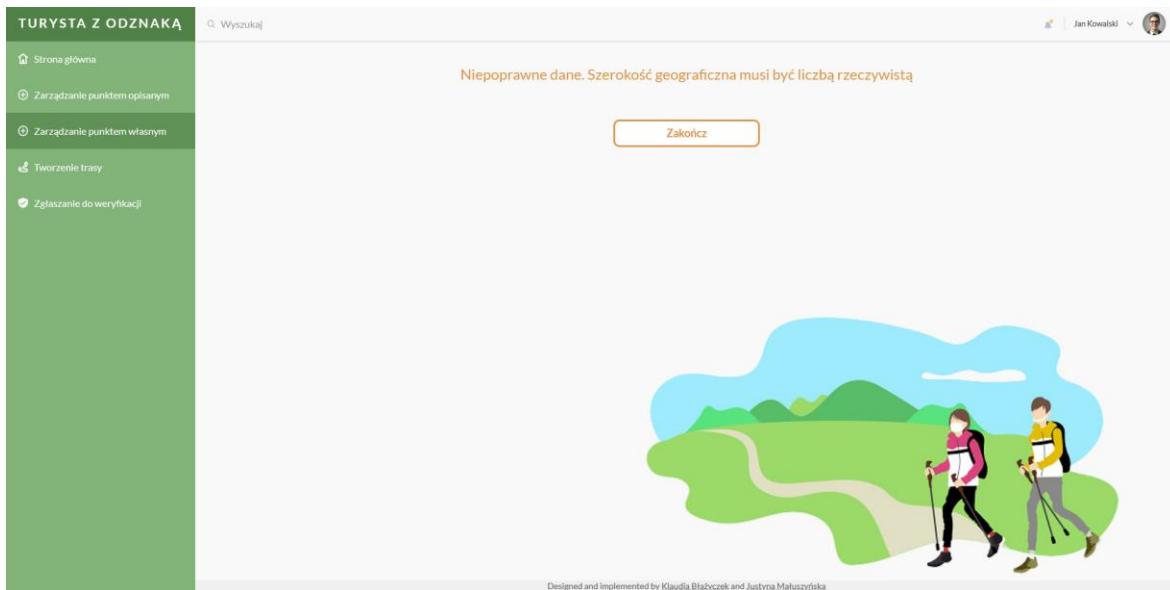
3. Znaczącą zmianą jest także odpowiednia implementacja na frontendzie, pozwalająca nam uniknąć występowania niektórych błędów, które zostały uwzględnione na mockupach. Przykładem jest tutaj wprowadzanie w formularzu danych dotyczących szerokości i długości geograficznej. Są to liczby rzeczywiste. Mockup uwzględniał błędy występujące po stronie klienta podczas wprowadzania tych wartości (np. poprzez podanie łańcucha). W implementacji do wprowadzania tych danych zastosowano sprawdzające wyrażenie regularne, które nie pozwala użytkownikowi wprowadzać innych znaków niż cyfry. Możliwe jest natomiast w dalszym ciągu wystąpienie błędu, w sytuacji gdy użytkownik poda nam jedynie liczbę naturalną:

Mockup:



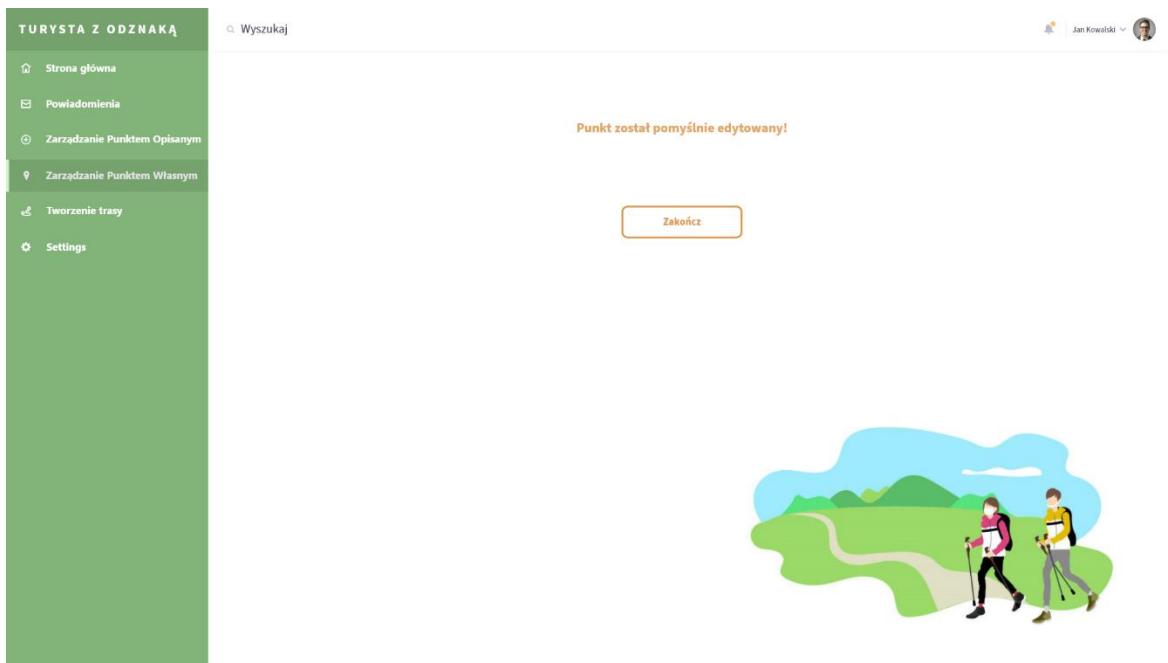
“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

Realizacja:



- Zakończenie wykonywania dowolnej akcji – nie ma znaczących zmian pomiędzy mockupem a realizacją. Zrezygnowano z pogrubienia czcionki, zamiast tego zastosowano większy rozmiar czcionki:

Mockup:



“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

Realizacja:

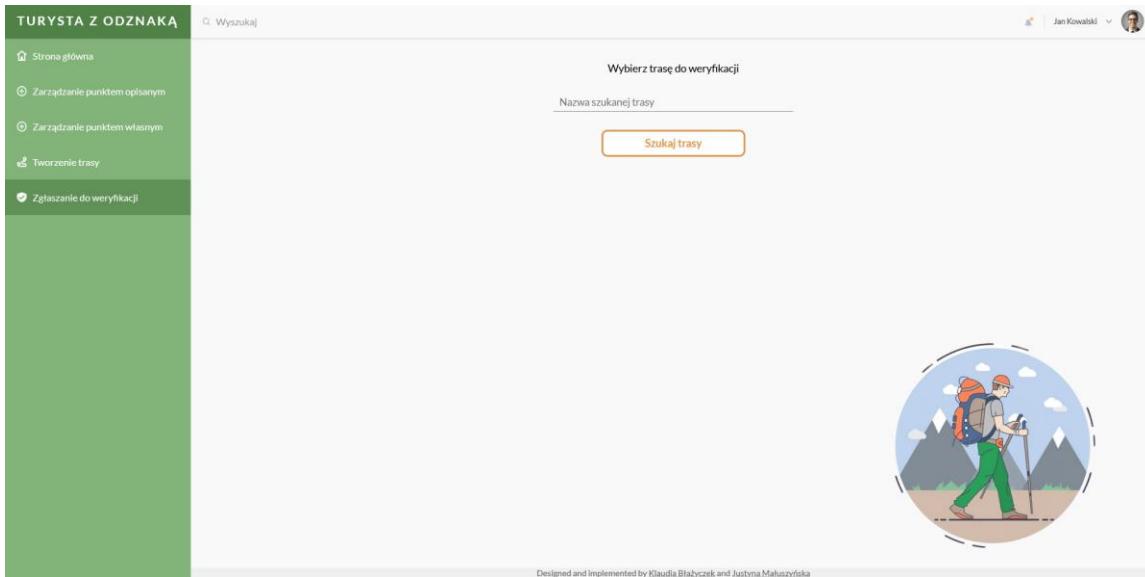
Przypadek użycia: Zgłaszcenie odcinków tras do weryfikacji (Justyna Maluszyńska)

Podobnie jak w poprzednich przypadkach użycia, zdecydowano się na zastosowanie podpowiedzi (placeholder) zamiast podpisów odpowiednich pól formularzy. Początek realizacji przypadku użycia nie odbiega poza tą zmianą od wyglądu mockupu:

Mockup:

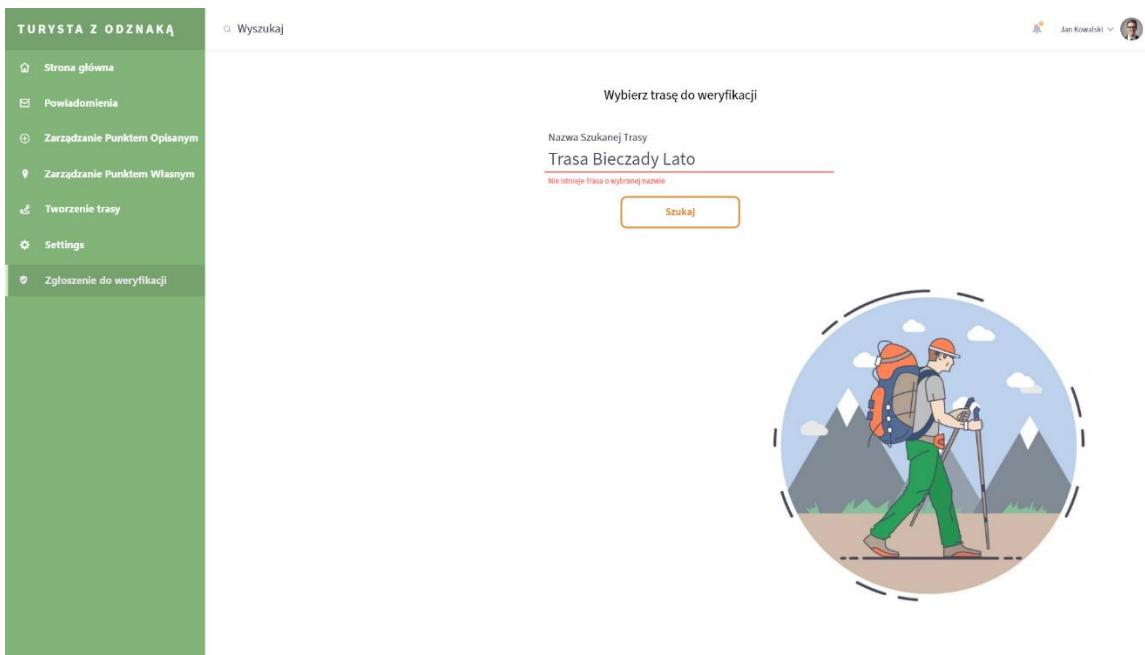
Realizacja:

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>



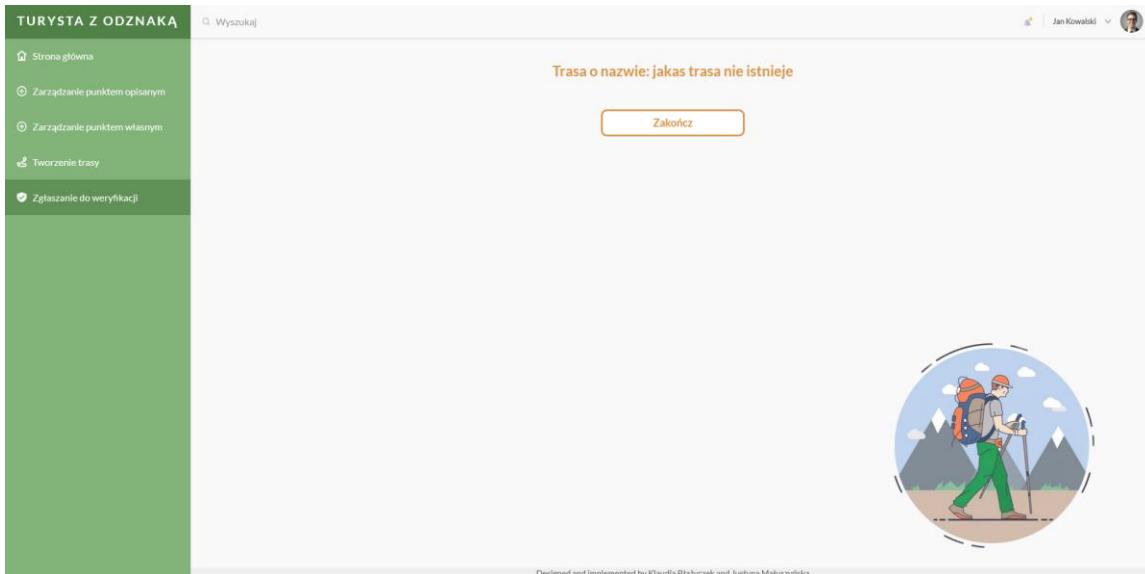
W przypadku niepoprawnej nazwy trasy (np. trasy która nie istnieje), zmieniono sposób wyświetlania błędu użytkownikowi. Finalnie zamiast czerwonych podpowiedzi, użytkownik dostaje komunikaty z błędem z czerwoną kolorze pomarańczowym. Zmieniono także niektóre komunikaty błędów, co nie wpływa znacząco na ogólny wygląd aplikacji.

Mockup:



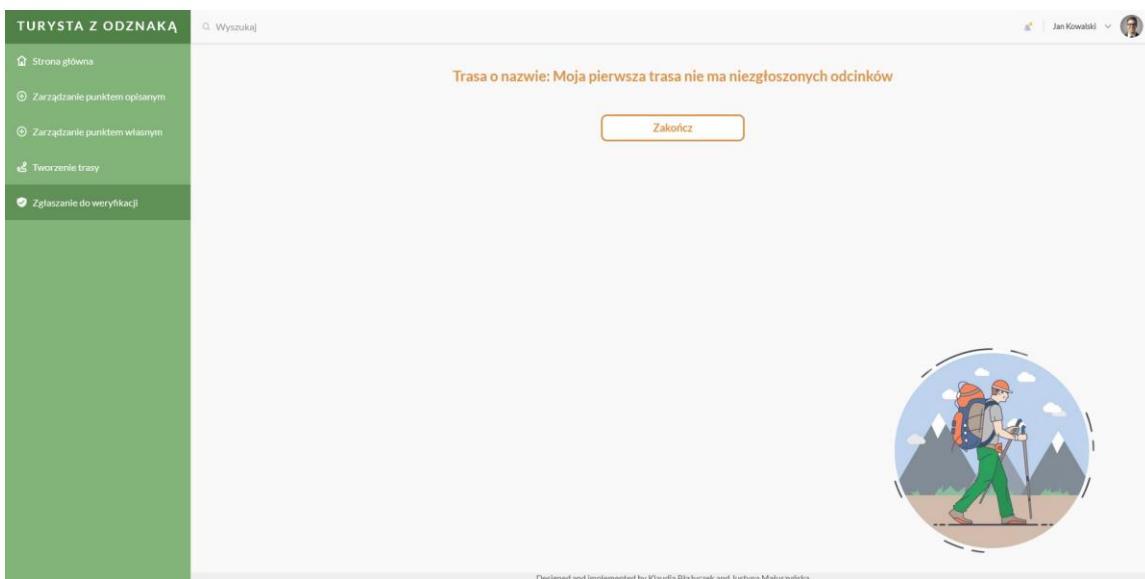
Realizacja:

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>



Podczas projektowania mockupu, nie przewidziano sytuacji, w której trasa może nie mieć już nie zgłoszonych odcinków do weryfikacji (np. jest już cała zatwierdzona albo oczekuje na zatwierdzenie). Dodano więc taki komunikat błędu na etapie realizacji:

Realizacja:



Podczas implementacji zrezygnowano również z wyświetlania liczby porządkowej podczas wyświetlania odcinków tras. Wprowadzało to zamieszanie, w przypadku gdy jakieś odcinki zostały już wcześniej zatwierdzone i numeracja nie była w takim przypadku pomocna.

Mockup:

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

TURYSTA Z ODZNAKA

Wyszukaj

Jan Kowalski

Wybierz odcinki do weryfikacji

Trasa Bieszczady Lato 2021

Lp.	POCZĘTEK	KONIEC	GRUPA GÓRSKA	LICZBA PUNKTÓW
1.	Wołosate	Przełęcz Bukowska	Bieszczady	8
2.	Przełęcz Bukowska	Rozsypaniec	Bieszczady	6
3.	Rozsypaniec	Halicz	Bieszczady	12
4.	Halicz	Przełęcz Goprowska	Bieszczady	8

Dalej

Dodane załączniki: 0
Przypisani weryfikujący: 0

Zgłoś

Realizacja:

TURYSTA Z ODZNAKA

Wyszukaj

Jan Kowalski

Wybierz odcinki do weryfikacji

Trasa testowa

Początek	Koniec	Grupa górska	Liczba punktów
Klimczok	Bystra Śląska	Beskid Śląski	12
Bystra Śląska	Przełęcz Kołowrót	Tatry Wysokie	4
Przełęcz Kołowrót	Bystra Śląska	Beskid Śląski	7

Dalej

Dodane załączniki: 0
Przypisani weryfikujący: 0

Zgłoś

Designed and implemented by Klaudia Błażecka and Justyna Małuszyska

Wybór odcinków do weryfikacji nie ma zmian w stosunku do mockupu:

Mockup:

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

TURYSTA Z ODZNAKĄ

Wyszukaj

Strona główna

Powiadomienia

Zarządzanie Punktem Opisanym

Zarządzanie Punktem Własnym

Tworzenie trasy

Settings

Zgłoszenie do weryfikacji

Wybierz odcinki do weryfikacji

Trasa Bieszczady Lato 2021

Lp.	POCZĄTEK	KONIEC	GRUPA GÓRSKA	LICZBA PUNKIÓW
1.	Wolosate	Przełęcz Bukowska	Bieszczady	8
2.	Przełęcz Bukowska	Rozsypaniec	Bieszczady	6
3.	Rozsypaniec	Halicz	Bieszczady	12
4.	Halicz	Przełęcz Goprowska	Bieszczady	8

Dodaj załącznik

Przypisz weryfikującego

Dodane załączniki: 0
Przypisani weryfikujących: 0

Zgłoś

Realizacja:

TURYSTA Z ODZNAKĄ

Wyszukaj

Strona główna

Zarządzanie punktem opisanym

Zarządzanie punktem własnym

Tworzenie trasy

Zgłoszenie do weryfikacji

Wybierz odcinki do weryfikacji

Trasa testowa

Początek	Koniec	Grupa górska	Liczba punktów
Klimczok	Bystra Śląska	Beskid Śląski	12
Bystra Śląska	Przełęcz Kołowrot	Tatry Wysokie	4
Przełęcz Kołowrot	Bystra Śląska	Beskid Śląski	7

Dalej

Dodane załączniki: 0
Przypisani weryfikujących: 0

Zgłoś

Nastąpiły zmiany w formularzu wprowadzania dat. Na mockupie nie przewidziano sposobu wprowadzania dat (wpisywanie ręczne albo wybór z kalendarza). Finalnie zdecydowano się na wprowadzanie dat wprost z wyskakującego kalendarza. Jest to łatwiejszy i bardziej intuicyjny sposób podawania dat. Co więcej, pomaga on przy walidacji i zapobiega wcześniejszym problemom przewidzianym na mockupie (np. data początkowa późniejsza niż data końcowa, wybór daty w której odcinek był nieczynny lub zamknięty, wybór daty z przeszłości).

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

Mockup:

The mockup shows a user interface for managing hiking trails. On the left is a green sidebar with a navigation menu:

- Strona główna
- Powiadomienia
- Zarządzanie Punktem Opisanym
- Zarządzanie Punktem Własnym
- Tworzenie trasy
- Settings
- Zgłoszenie do weryfikacji

The main area displays two trail segments with start and end dates:

- 1. Wołosate -> Przełęcz Bukowska: Data rozpoczęcia 03.07.2021, Data zakończenia 03.07.2021
- 3. Schronisko w Dolinie Roztoki -> Wodogrzmoty Mickiewicza: Data rozpoczęcia 04.07.2021, Data zakończenia 04.07.2021

A large orange "Zapisz daty" button is centered below the segments. To the right is a circular illustration of a hiker walking through a landscape.

Realizacja:

The realization shows a similar interface to the mockup. The sidebar includes:

- Strona główna
- Zarządzanie punktem opisanym
- Zarządzanie punktem własnym
- Tworzenie trasy
- Zgłoszenie do weryfikacji

The main area shows two trail segments and a calendar for selecting dates:

- Bystra Śląska -> Przełęcz Kołowrót: Data rozpoczęcia, Data zakończenia
- Przełęcz Kołowrót -> Bystra Śląska: Data rozpoczęcia

A calendar for January 2022 is displayed, with the 14th highlighted in blue. An orange "Zapisz" button is positioned next to the calendar. To the right is a circular illustration of a hiker.

Designed and implemented by Klaudia Błażczyk and Justyna Małuszyska

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

Zadbano również o konieczność wypełnienia wszystkich pól formularza, dlatego w przypadku, gdy użytkownik nie poda wszystkich danych, podawany jest mu stosowny komunikat w kolorze czerwonym.

Realizacja:

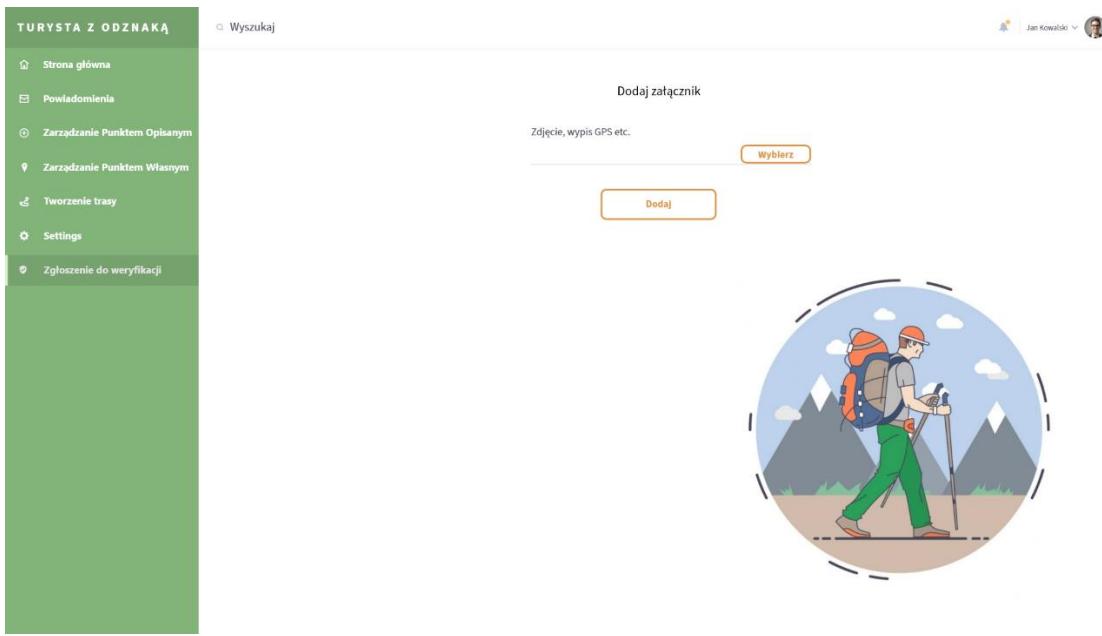
W polu wprowadzania numeru legitymacji przewodnika, zadbano w odpowiedni sposób od strony frontendowej o poprawność wprowadzanych danych poprzez wyrażenie regularne. Pomaga to ochronie przed niechcianymi błędami spowodowanymi niepoprawnymi danymi wprowadzonymi przez użytkownika. W momencie, gdy użytkownik podaje dane nieistniejącego przewodnika, wyświetlany jest mu stosowny komunikat:

Realizacja:

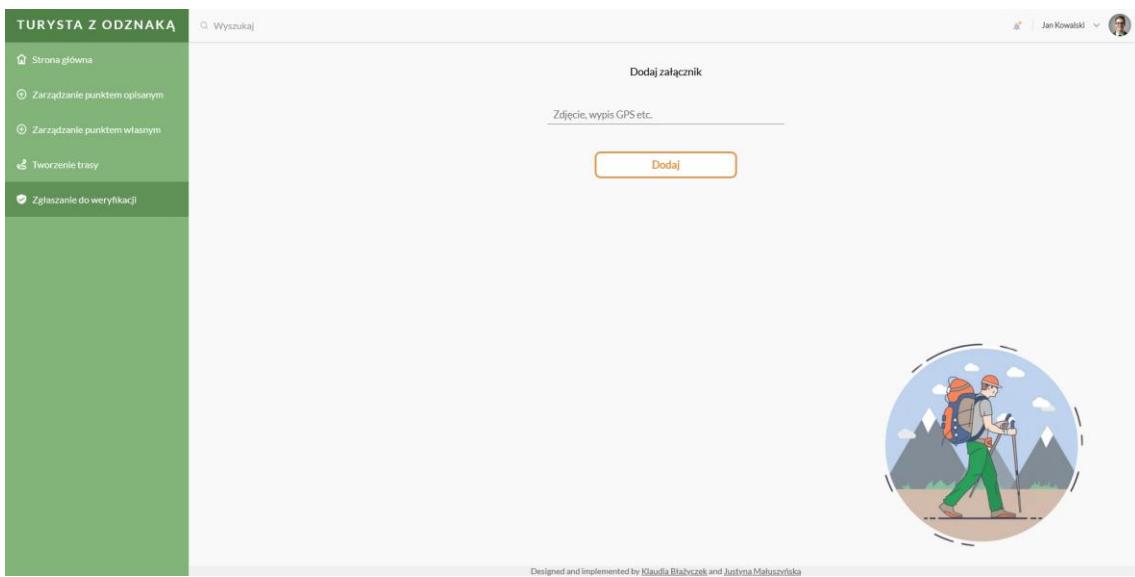
“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

Drobne zmiany nastąpiły w miejscu wprowadzania załącznika dowodu (np. fotografia w formacie .png albo wypis gps w formacie .pdf). Poprzez uzgodnione z prowadzącym uproszczenie, zdecydowano się na zwykłe pole przyjmujące łańcuch znaków, a nie jak proponowano w mockupie przycisk do fizycznego wybioru załącznika. Jest to pewnego rodzaju symulacja wprowadzania tam faktycznych załączników, spowodowana również po części uproszczoną architekturą naszego systemu.

Mockup:



Realizacja:



“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

Ostatnią kluczowym dodatkiem w stosunku do mockupów, jest dodanie podsumowania po pomyślnym zapisaniu i zgłoszeniu wybranych odcinków tras do weryfikacji:

Realizacja:

Początek	Koniec	Grupa górskie	Start	Koniec	Przewodnik	Załącznik
Bystra Śląska	Przełęcz Kołowrót	Tatry Wysokie	2021-12-26	2021-12-26	-	plik.png
Przełęcz Kołowrót	Bystra Śląska	Beskid Śląski	2022-01-03	2022-01-03	123456	-
Klimczok	Bystra Śląska	Beskid Śląski	2022-01-06	2022-01-06	123456	-

Testowanie

1. Testy systemowe – przypadki/procedury testowe

Zgodnie z ustaleniami z prowadzącym, poniżej przedstawiamy kilka przykładowych procedur testowych dla każdego z przypadków użycia. Procedury dotyczą zarówno poprawnego działania całej logiki systemu, jak i poprawnej interakcji z użytkownikiem (poprawne renderowanie widoków, wyświetlanie odpowiednich wyników, wiadomości błędów). Testy dotyczą sytuacji typowych (common case), ale testowane są także niektóre przykładowe sytuacje brzegowe.

1.1 Zarządzanie Punktem Własnym (Justyna Małuszyńska)

TC ID	Idea testowa	Kroki testu	Punkty kontrolne	Dane testowe	Oczekiwane wyniki
PT001	Procedura testowa dla przypadku użycia <i>Dodawanie Punktu Własnego</i> . Weryfikacja poprawnego utworzenia strony i przenoszenia na odpowiedni adres URL	1. Użytkownik wybiera opcję „Dodaj punkt własny” 2. System przenosi na inny adres URL	Krok 0: Sprawdzić, czy system daje możliwość wybrania opcji dodawania punktu Krok 2: Sprawdzić, czy system przeniósł na poprawny adres URL	brak	Wyświetlenie formularza dodawania nowego punktu.

TC ID	Idea testowa	Kroki testu	Punkty kontrolne	Dane testowe	Oczekiwane wyniki
PT002	Procedura testowa dla przypadku użycia <i>Dodawanie Punktu Własnego</i> . Weryfikacja poprawności realizacji PU dla poprawnych wymaganych danych.	1. Użytkownik wybiera opcję „Dodaj punkt własny” (PT001) 2. System wyświetla formularz i żąda danych nowego punktu 3. Użytkownik podaje dane punktu 4. Użytkownik wybiera opcję „Dodaj”	Krok 2: Sprawdzić, czy system umożliwia wprowadzanie danych Krok 6: Sprawdzić, czy potwierdzenie dodania punktu jest	Nazwa: <i>Nowy punkt własny</i> Szerokość geograficzna: <i>45.6756</i> Długość geograficzna: <i>43.4342</i>	Dodanie punktu do bazy. Wyświetlenie potwierdzenia dodania punktu <i>Punkt Nowy punkt własny został pomyślnie dodany</i> .

“Turysta Z Odznaką”<Project Name>		
Etap I		Data: <dd/mmm/yy>

		5. System stwierdza poprawność danych. Dodaje punkt własny do bazy danych 6. System wyświetla potwierdzenie dodania punktu	widoczne i zgodne z oczekiwanyymi wynikami	Baza danych przed testem nie zawiera punktu z wyżej wymienionymi danymi.	Oczekiwane wyniki
TC ID	Idea testowa	Kroki testu	Punkty kontrolne	Dane testowe	Oczekiwane wyniki
PT003	Procedura testowa dla przypadku użycia <i>Dodawanie Punktu Własnego</i> . Weryfikacja poprawności realizacji PU dla niepoprawnych wymaganych danych.	1. Użytkownik wybiera opcję „Dodaj punkt własny” (PT001) 2. System wyświetla formularz i żąda danych nowego punktu 3. Użytkownik podaje dane punktu 4. Użytkownik wybiera opcję „Dodaj” 5. System stwierdza niepoprawność danych. 6. System wyświetla informację o błędzie (nieunikalna nazwa)	Krok 2: Sprawdzić, czy system umożliwia wprowadzanie danych Krok 6: Sprawdzić, czy system wyświetla informację o błędzie z odpowiednim komunikatem	Nazwa (nieunikalna): <i>Lysica</i> Szerokość geograficzna: 45.6756 Długość geograficzna: 43.4342 Baza danych przed testem zawiera punkt z nadaną nazwą <i>Lysica</i> .	Wyświetlenie informacji o błędzie <i>Punkt o takiej nazwie już istnieje. Wybierz inną nazwę</i> .

TC ID	Idea testowa	Kroki testu	Punkty kontrolne	Dane testowe	Oczekiwane wyniki
PT004	Procedura testowa dla przypadku użycia <i>Dodawanie Punktu Własnego</i> . Weryfikacja poprawności realizacji PU dla niewystarczającej ilości danych.	1. Użytkownik wybiera opcję „Dodaj punkt własny” (PT001) 2. System wyświetla formularz i żąda danych nowego punktu 3. Użytkownik podaje nazwę punktu 4. Użytkownik wybiera opcję „Dodaj” 5. System stwierdza niewystarczającą ilość danych. Wyświetla informację o błędzie 6. Użytkownik podaje szerokość geograficzną punktu 7. Użytkownik wybiera opcję „Dodaj” 8. System stwierdza niewystarczającą ilość danych. Wyświetla informację o błędzie 9. Użytkownik podaje długość geograficzną punktu 10. Użytkownik wybiera opcję „Dodaj” 11. System stwierdza poprawność danych. Dodaje punkt własny do bazy danych 12. System wyświetla potwierdzenie dodania punktu	Krok 2: Sprawdzić, czy system umożliwia wprowadzanie danych Krok 5: Sprawdzić, czy system podał informację o błędzie nr 1 Krok 8: Sprawdzić, czy system podał informację o błędzie nr 2 Krok 12: Sprawdzić, czy potwierdzenie dodania punktu jest widoczne i zgodne z oczekiwanyymi wynikami	Nazwa: <i>Jakaś nazwa</i> Szerokość geograficzna: 12.3456 Długość geograficzna: 45.5784 Baza danych przed testem nie zawiera punktu z wyżej wymienionymi danymi.	Informowanie użytkownika na bieżąco o braku wymaganych danych. Błąd nr 1: <i>Nie podano szerokości geograficznej punktu własnego</i> Błąd nr 2: <i>Nie podano długości geograficznej punktu własnego</i> Potwierdzenie dodania punktu: <i>Punkt Jakaś nazwa został pomyślnie dodany</i> .

1.2 Zgłaszanie prośby o zweryfikowanie Trasy (Justyna Małuszyńska)

TC ID	Idea testowa	Kroki testu	Punkty kontrolne	Dane testowe	Oczekiwane wyniki
-------	--------------	-------------	------------------	--------------	-------------------

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

PT005	<p>Procedura testowa dla przypadku użycia <i>Zgłaszanie prośby o zweryfikowanie trasy</i>. Weryfikacja poprawnego utworzenia strony i wyświetlania odcinków trasy w formie tabeli</p>	<ol style="list-style-type: none"> 1. Użytkownik wybiera opcję „Zgłaszanie do weryfikacji” 2. System wyświetla formularz i żąda nazwy szukanej trasy, podpowiadając przy tym za pomocą Comboboxa 3. Użytkownik wprowadza słowo kluczowe i wybiera podpowiadaną trasę 4. System przetwarza trasę i stwierdza poprawność danych 5. System wyświetla listę odcinków trasy w formie listy 	<p>Krok 2: Sprawdzić, czy system umożliwia wprowadzenie danych i podpowiada wartości</p> <p>Krok 5: Sprawdzić, czy system wyświetli wszystkie odcinki wchodzące w skład trasy</p>	<p>Słowo kluczowe: <i>moja</i> Wybrana podpowiedź: <i>Moja pierwsza trasa</i></p>	<p>Wyświetlenie formularza wpisywania nazwy trasy, wyświetlenie odcinków szukanej trasy (m.in. <i>Schronisko PTTK Hala Szrenicka, Szrenica, Przełęcz Mokra, Wodospad Kamieńczyk, Śnieżne Kotły</i>)</p>
-------	---	--	---	---	---

TC ID	Idea testowa	Kroki testu	Punkty kontrolne	Dane testowe	Oczekiwane wyniki
PT006	<p>Procedura testowa dla przypadku użycia <i>Zgłaszanie prośby o zweryfikowanie trasy</i>. Weryfikacja poprawnego utworzenia formularzy wprowadzania dat i wyboru odcinków tras</p>	<ol style="list-style-type: none"> 1. Użytkownik wybiera opcję „Zgłaszanie do weryfikacji” i wprowadza nazwę szukanej trasy 2. System stwierdza poprawność danych i wyświetla listę odcinków (PT005) 3. Użytkownik wybiera odcinki i naciska przycisk „Dalej” 4. System stwierdza poprawność danych i wyświetla formularze do wprowadzenia dat 5. Użytkownik wprowadza daty początku i końca każdego z odcinków 6. System sprawdza, czy wszystkie daty zostały wybrane i wyświetla listę odcinków z możliwością wyboru rodzaju dowodu 	<p>Krok 4: Sprawdzić, czy system wyświetli odpowiednią ilość formularzy do wprowadzania dat</p> <p>Krok 6: Sprawdzić, czy system odpowiednio przetworzył wygląd strony (zmiana backgroundu wybranych odcinków) i dał możliwość wyboru rodzaju dowodu</p>	<p>Słowo kluczowe: <i>moja</i> Wybrana podpowiedź: <i>Moja pierwsza trasa</i> Wybrane dwa pierwsze odcinki trasy.</p>	<p>Umożliwienie wyboru odcinków, wybrania dat i przygotowanie strony do dalszych kroków</p>

TC ID	Idea testowa	Kroki testu	Punkty kontrolne	Dane testowe	Oczekiwane wyniki
PT007	<p>Procedura testowa dla przypadku użycia <i>Zgłaszanie prośby o zweryfikowanie trasy</i>. Weryfikacja poprawnego utworzenia formularza wprowadzania błędnego numeru legitymacji przewodnika oraz obsługa błędów</p>	<ol style="list-style-type: none"> 1. Użytkownik wybiera opcję „Zgłaszanie do weryfikacji” i wprowadza nazwę szukanej trasy 2. System stwierdza poprawność danych i wyświetla listę odcinków (PT005) 3. Użytkownik wybiera odcinki i wprowadza daty (PT006) 4. System sprawdza, czy wszystkie daty zostały wybrane i wyświetla listę odcinków z możliwością wyboru rodzaju 	<p>Krok 6: Sprawdzić, czy system odpowiednio wygenerował widok strony</p> <p>Krok 8: Sprawdzić, czy system zareagował na błąd i zwrócił odpowiedni komunikat</p>	<p>Słowo kluczowe: <i>moja</i> Wybrana podpowiedź: <i>Moja pierwsza trasa</i> Wybrane dwa pierwsze odcinki trasy. Numer legitymacji przewodnika: <i>654321</i></p>	<p>Odpowiednia weryfikacja i obsługa błędów związanych z numerem legitymacji przewodnika. Zwrócenie stosownego komunikatu: <i>Przewodnik o podanym numerze legitymacji nie</i></p>

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

	dowodu 5. Użytkownik wybiera opcję „Przypisz przewodnika” 6. System wyświetla formularz wprowadzania numeru legitymacji 7. Użytkownik wprowadza numer legitymacji 8. System stwierdza wystąpienie błędu oraz informuje użytkownika stosownym komunikatem				istnieje
--	--	--	--	--	----------

1.3 Zarządzanie punktem opisanym (Klaudia Błażyczek)

TC ID	Idea testowa	Kroki testu	Punkty kontrolne	Dane testowe	Oczekiwane wyniki
PT008	Procedura testowa dla przypadku użycia <i>Dodawanie Punktu Opisanego</i> . Weryfikacja poprawnego utworzenia strony głównej zarządzania punktem opisanym. Jest to niepełny przypadek – ma on na celu jedynie sprawdzić poprawność generowania całego menu zarządzania punktem opisanym	1. Użytkownik wybiera opcję „Zarządzaj punktem opisanym” 2. System przenosi na inny adres URL	Krok 2: Sprawdzić, czy system przeniósł na poprawny adres URL zarządzania punktem opisanym oraz czy zawartość menu zawiera widoczne przyciski z odpowiednimi nazwami, które umożliwiają przejście do poszczególnych akcji CRUD'a.	brak	Wyświetlenie menu zarządzania punktem opisanym

TC ID	Idea testowa	Kroki testu	Punkty kontrolne	Dane testowe	Oczekiwane wyniki
PT009	Procedura testowa dla przypadku użycia <i>Dodawanie Punktu Opisanego</i> . Weryfikacja poprawności realizacji PU dla poprawnych wymaganych danych.	1. Użytkownik wybiera opcję „Zarządzaj punktem opisanym” 2. System przenosi na adres URL zarządzania punktem opisanym 3. Użytkownik wybiera opcję „Dodaj punkt opisany” 4. System przenosi na inny adres URL oraz wyświetla formularz i żąda danych nowego punktu 5. Użytkownik podaje dane punktu 6. Użytkownik zatwierdza formularz poprzez naduszenie przycisku „Dodaj punkt opisany” 7. System stwierdza	Krok 4: Sprawdzić, czy przeniesiono na poprawny adres URL oraz sprawdzić, czy system umożliwia wprowadzanie danych Krok 6: Sprawdzić, czy przycisk potwierdzający dodanie punktu jest widoczny, a jego tekst zgodny z oczekiwany Krok 8: Sprawdzić, czy komunikat wyświetlany użytkownikowi jest	Nazwa: Zielone Berdo Wysokość: 989 Baza danych przed testem nie zawiera punktu z wyżej wymienionymi danymi.	Dodanie punktu do bazy. Wyświetlenie potwierdzenia dodania punktu <i>Punkt Zielone Berdo został pomyślnie dodany</i>

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

		poprawność danych. Dodaje punkt własny do bazy danych 8. System wyświetla potwierdzenie dodania punktu i wyświetla przycisk „Zakończ” do powrotu do menu 9. Użytkownik kliką przycisk	taki jak oczekiwano oraz czy widoczny jest przycisk „Zakończ”		
--	--	---	---	--	--

TC ID	Idea testowa	Kroki testu	Punkty kontrolne	Dane testowe	Oczekiwane wyniki
PT010	Procedura testowa dla przypadku użycia <i>Dodawanie Punktu Opisanego</i> . Weryfikacja poprawności realizacji PU dla niepoprawnych danych.	1. Użytkownik wybiera opcję „Zarządzaj punktem opisanym” 2. System przenosi na adres URL zarządzania punktem opisanym 3. Użytkownik wybiera opcję „Dodaj punkt opisany” 4. System przenosi na inny adres URL oraz wyświetla formularz i żąda danych nowego punktu 5. Użytkownik podaje dane punktu 6. Użytkownik zatwierdza formularz poprzez naduszenie przycisku „Dodaj punkt opisany” 7. System stwierdza niepoprawność danych. 8. System wyświetla informacje o nieunikalnej nazwie i wyświetla przycisk „Zakończ” do powrotu do menu 9. Użytkownik kliką przycisk	Krok 4: Sprawdzić, czy przeniesiono na poprawny adres URL oraz sprawdzić, czy system umożliwia wprowadzanie danych Krok 6: Sprawdzić, czy przycisk potwierdzający dodanie punktu jest widoczny, a jego tekst zgodny z oczekiwanyimi Krok 8: Sprawdzić, czy komunikat wyświetlany użytkownikowi jest taki jak oczekiwano oraz czy widoczny jest przycisk „Zakończ”	Nazwa: <i>weTliNa</i> Baza danych przed testem zawiera punkt Wetlina	Wyświetlenie informacji o błędzie <i>Punkt o takiej nazwie już istnieje. Wybierz inną nazwę</i>

TC ID	Idea testowa	Kroki testu	Punkty kontrolne	Dane testowe	Oczekiwane wyniki
PT011	Procedura testowa dla przypadku użycia <i>Edycja Punktu Opisanego</i> . Weryfikacja poprawności realizacji PU dla niepoprawnych danych (próba edycji punktu opisanego, który nie istnieje).	1. Użytkownik wybiera opcję „Zarządzaj punktem opisanym” 2. System przenosi na adres URL zarządzania punktem opisanym 3. Użytkownik wybiera opcję „Edytuj punkt opisany” 4. System przenosi na inny adres URL oraz wyświetla formularz do wpisania/wybrania nazwy punktu do edycji 5. Użytkownik podaje nazwę punktu 6. Użytkownik zatwierdza formularz poprzez naduszenie przycisku „Edytuj punkt opisany” 7. System stwierdza	Krok 4: Sprawdzić, czy przeniesiono na poprawny adres URL oraz sprawdzić, czy system umożliwia wprowadzanie danych Krok 6: Sprawdzić, czy przycisk potwierdzający edycję punktu jest widoczny, a jego tekst zgodny z oczekiwanyimi Krok 8: Sprawdzić, czy komunikat wyświetlany użytkownikowi jest	Nazwa: <i>hdfhd</i> Baza danych przed testem nie zawiera punktu opisanego <i>hdfhd</i>	Wyświetlenie informacji o błędzie <i>Punkt opisany o nazwie: hdfhd nie istnieje</i>

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

		niepoprawność danych. 8. System wyświetla informację o tym, że punkt nie istnieje i wyświetla przycisk „Zakończ” do powrotu do menu 9. Użytkownik kliką przycisk	taki jak oczekiwano oraz czy widoczny jest przycisk „Zakończ”		
--	--	--	---	--	--

TC ID	Idea testowa	Kroki testu	Punkty kontrolne	Dane testowe	Oczekiwane wyniki
PT012	Procedura testowa dla przypadku użycia <i>Edycja Punktu Opisanego</i> . Weryfikacja poprawności realizacji PU dla niepoprawnych danych (próba edycji punktu opisanego, który jest używany w odcinkach).	1. Użytkownik wybiera opcję „Zarządzaj punktem opisanym” 2. System przenosi na adres URL zarządzania punktem opisanym 3. Użytkownik wybiera opcję „Edytuj punkt opisany” 4. System przenosi na inny adres URL oraz wyświetla formularz do wpisania/wybrania nazwy punktu do edycji 5. Użytkownik podaje nazwę punktu 6. Użytkownik zatwierdza formularz poprzez naduszenie przycisku „Edytuj punkt opisany” 7. System stwierdza, że punkt nie może być edytowany. 8. System wyświetla informację o tym, że punkt nie może być edytowany i wyświetla przycisk „Zakończ” do powrotu do menu 9. Użytkownik kliką przycisk	Krok 4: Sprawdzić, czy przeniesiono na poprawny adres URL oraz sprawdzić, czy system umożliwia wprowadzanie danych Krok 6: Sprawdzić, czy przycisk potwierdzający edycję punktu jest widoczny, a jego tekst zgodny z oczekiwanyimi Krok 8: Sprawdzić, czy komunikat wyświetlany użytkownikowi jest taki jak oczekiwano oraz czy widoczny jest przycisk „Zakończ”	Nazwa: <i>Wetlina</i> Baza danych przed testem zawiera punkt <i>Wetlina</i> , a punkt jest używany w odcinkach	Wyświetlenie informacji o błędzie <i>Punkt Wetlina jest już używany i nie można go edytować</i>

TC ID	Idea testowa	Kroki testu	Punkty kontrolne	Dane testowe	Oczekiwane wyniki
PT013	Procedura testowa dla przypadku użycia <i>Edycja Punktu Opisanego</i> . Weryfikacja poprawności realizacji PU dla niepoprawnych danych (próba edycji punktu opisanego i nadanie mu istniejącej nazwy).	1. Użytkownik wybiera opcję „Zarządzaj punktem opisanym” 2. System przenosi na adres URL zarządzania punktem opisanym 3. Użytkownik wybiera opcję „Edytuj punkt opisany” 4. System przenosi na inny adres URL oraz wyświetla formularz do wpisania/wybrania nazwy punktu do edycji 5. Użytkownik podaje nazwę punktu 6. Użytkownik zatwierdza formularz poprzez naduszenie przycisku „Edytuj punkt opisany” 7. System stwierdza, że punkt	Krok 4: Sprawdzić, czy przeniesiono na poprawny adres URL oraz sprawdzić, czy system umożliwia wprowadzanie danych Krok 6: Sprawdzić, czy przycisk potwierdzający edycję punktu jest widoczny, a jego tekst zgodny z oczekiwanyimi Krok 10: Sprawdzić, czy przycisk potwierdzający edycję punktu jest	Nazwa edytowanego punktu: <i>Kozi Wierch</i> Nowa nazwa punktu: <i>Wetlina</i> Baza danych przed testem zawiera punkt <i>Wetlina</i> oraz <i>Kozi Wierch</i>	Wyświetlenie informacji o błędzie <i>Punkt o takiej nazwie już istnieje. Wybierz inną nazwę</i>

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

	<p>może być edytowany</p> <p>8. System wyświetla formularz do wpisania nowej nazwy i opcjonalnie wysokości punktu</p> <p>9. Użytkownik podaje nowe dane</p> <p>10. Użytkownik zatwierdza formularz poprzez naduszenie przycisku „Edytuj punkt opisany”</p> <p>11. System stwierdza, że nie można nadać punktowi takiej nazwy, gdyż taki już istnieje.</p> <p>12. System wyświetla informację o tym, że punkt o takiej nazwie już istnieje i nie można tak nazwać punktu i wyświetla przycisk „Zakończ” do powrotu do menu</p> <p>13. Użytkownik kliką przycisk</p>	widoczny, a jego tekst zgodny z oczekiwanyimi	Krok 12: Sprawdzić, czy komunikat wyświetlany użytkownikowi jest taki jak oczekiwano oraz czy widoczny jest przycisk „Zakończ”	
--	--	---	--	--

1.4 Tworzenie trasy (Klaudia Błażczek)

TC ID	Idea testowa	Kroki testu	Punkty kontrolne	Dane testowe	Oczekiwane wyniki
PT014	Procedura testowa dla przypadku użycia <i>Tworzenie trasy</i> . Weryfikacja poprawności realizacji PU dla poprawnych danych (dane zawsze będą poprawne)	<p>1. Użytkownik wybiera opcję „Tworzenie trasy”</p> <p>2. System przenosi na adres URL tworzenia trasy oraz wyświetla formularz do podania punktu początkowego</p> <p>3. Użytkownik nie wpisuje nazwy punktu początkowego i zatwierdza formularz</p> <p>4. System sprawdza, że nie podano punktu początkowego i wyświetla informację o błędzie</p> <p>5. Użytkownik wpisuje nieistniejący punkt i zatwierdza formularz</p> <p>6. System sprawdza, że podany punkt początkowy nie istnieje i wyświetla informację o błędzie</p> <p>7. Użytkownik wpisuje poprawny punkt i zatwierdza formularz</p> <p>8. System sprawdza, że podany punkt istnieje i wyświetla formularz do wybierania kolejnych punktów trasy</p> <p>9. Użytkownik z listy wybiera kolejne dwa punkty trasy</p> <p>10. Użytkownik rozmyśla się i po kolei usuwa wszystkie</p>	<p>Krok 2: Sprawdzić, czy przeniesiono na poprawny adres URL oraz sprawdzić, czy system poprawnie wytworzył formularz z przyciskiem oraz licznikiem punktów GOT</p> <p>Krok 4: Sprawdzić, czy komunikat wyświetlany użytkownikowi jest taki jak oczekiwano</p> <p>Krok 6: Sprawdzić, czy komunikat wyświetlany użytkownikowi jest taki jak oczekiwano</p> <p>Krok 8: Sprawdzić, czy system poprawnie wytworzył formularz z przyciskami oraz licznikiem GOT</p> <p>Krok 15: Sprawdzić, czy komunikat wyświetlany użytkownikowi jest</p>	<p>Nazwa nieistniejącego punktu: <i>cvbvbcb</i></p> <p>Nazwa istniejącego punktu (pierwsza próba): <i>Polanki</i></p> <p>Nazwy kolejnych punktów(dla pierwszej próby): <i>Durna, Polanki</i></p> <p>Nazwa istniejącego punktu (dla drugiej próby): <i>Hotel PTTK Ustrzyki Górne</i></p> <p>Nazwy kolejnych punktów (dla drugiej próby): <i>Szeroki Wierch</i></p> <p>Istniejąca nazwa trasy: <i>Bieszczady 2022</i></p> <p>Nieistniejąca nazwa trasy: <i>Bieszczady z rodziną 2022</i></p> <p>Baza danych przed testem zawiera punkt: <i>Polanki, Durna, Hotel PTTK Ustrzyki Górne,</i></p>	<p>Informacja o wykonanej akcji: <i>Zapisano trasę: Bieszczady z rodziną 2022</i></p> <p>Informacja o punktacji: <i>Liczba punktów do GOT: 13</i></p> <p>Informacja o odcinkach trasy: 1.p.: <i>1, początek: Hotel PTTK Ustrzyki Górne, przez: -, koniec: Szeroki Wierch, grupa górska: Bieszczady, punkty: 13</i></p>

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

	<p>dodane punkty</p> <p>11. W momencie, gdy użytkownik usuwa punkt początkowy, system przenosi go z powrotem do formularza, w którym trzeba podać punkt początkowy</p> <p>12. Użytkownik podaje inny punkt początkowy</p> <p>13. System sprawdza, że podany punkt istnieje i wyświetla formularz do wybierania kolejnych punktów trasy</p> <p>14. Użytkownik zatwierdza formularz bez podania kolejnych punktów trasy</p> <p>15. System wyświetla informację o błędzie</p> <p>16. Użytkownik wybiera z listy kolejne punkty trasy</p> <p>17. Użytkownik zatwierdza wybrane punkty trasy</p> <p>18. System przenosi użytkownika do formularza nadawania trasie nazwy</p> <p>19. Użytkownik zatwierdza formularz bez podania nazwy trasy</p> <p>20. System sprawdza, że nazwa jest pusta i wyświetla odpowiedni komunikat</p> <p>21. Użytkownik wpisuje nazwę trasy, która już istnieje (turysta ma trasę o takiej nazwie)</p> <p>22. System sprawdza, że trasa o podanej nazwie już istnieje i wyświetla odpowiedni komunikat</p> <p>23. Użytkownik wpisuje nazwę trasy, która jeszcze nie istnieje (turysta nie ma trasy o takiej nazwie)</p> <p>24. System sprawdza, że trasa o podanej nazwie nie istnieje. Zapisuje trasę oraz odcinki trasy do bazy danych</p> <p>25. System wyświetla podsumowanie trasy wycieczki turyście (wybrane odcinki i informacje na ich temat, punktację GOT oraz nazwę trasy) i wyświetla przycisk „Zakończ” do powrotu do menu</p>	<p>taki jak oczekiwano</p> <p>Krok 16: Sprawdzenie, czy punkty GOT poprawnie się nalicyły</p> <p>Krok 18: Sprawdzić, czy system poprawnie wytworzył formularz z przyciskiem</p> <p>Krok 20: Sprawdzić, czy komunikat wyświetlany użytkownikowi jest taki jak oczekiwano</p> <p>Krok 22: Sprawdzić, czy komunikat wyświetlany użytkownikowi jest taki jak oczekiwano</p> <p>Krok 25: Sprawdzić, czy w podsumowaniu pojawiły się oczekiwane dane oraz czy widoczny jest przycisk „Zakończ”</p>	<p><i>Szeroki Wierch</i> oraz trasę <i>Bieszczady</i> 2022</p>	
--	--	--	--	--

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

	26. Użytkownik kliką przycisk	
--	-------------------------------	--

2. Testy systemowe automatyczne

Wszystkie wyżej wymienione przypadki testowe zostały zautomatyzowane. Wykorzystane zostało do tego narzędzie: Cypress. Testy automatyczne znajdują się w pakiecie: frontend/cypress/integration

W moim przypadku (Klaudia Błażyczek) IDE miało problem z poprawną identyfikacją narzędzia Cypress, dlatego na screenach, polecenia są podkreślone na czerwono, pomimo, że testy działają poprawnie – proszę ignorować te podkreślenia.

2.1 Zarządzanie punktem opisany (Klaudia Błażyczek)

```
describe("labeled points add e2e tests", () => {
  beforeEach(() => {
    cy.visit("/punkt-opisany");
  });

  it("checks if menu renders correctly", () => {
    cy.get("#buttonContainer").should("exist");
    cy.get(':nth-child(1) > a > .LinkButton_linkText_2AiYn').should('exist');
    cy.get(':nth-child(1) > a > .LinkButton_linkText_2AiYn').should('be.visible');
    cy.get(':nth-child(1) > a > .LinkButton_linkText_2AiYn').should('have.text', 'Dodaj punkt opisany');
    cy.get(':nth-child(2) > a > .LinkButton_linkText_2AiYn').should('exist');
    cy.get(':nth-child(2) > a > .LinkButton_linkText_2AiYn').should('be.visible');
    cy.get(':nth-child(2) > a > .LinkButton_linkText_2AiYn').should('have.text', 'Edytuj punkt opisany');
    cy.get(':nth-child(3) > a > .LinkButton_linkText_2AiYn').should('exist');
    cy.get(':nth-child(3) > a > .LinkButton_linkText_2AiYn').should('be.visible');
    cy.get(':nth-child(3) > a > .LinkButton_linkText_2AiYn').should('have.text', 'Usuń punkt opisany');
    cy.get(':nth-child(4) > a > .LinkButton_linkText_2AiYn').should('exist');
    cy.get(':nth-child(4) > a > .LinkButton_linkText_2AiYn').should('be.visible');
    cy.get(':nth-child(4) > a > .LinkButton_linkText_2AiYn').should('have.text', 'Szukaj punktów opisanych');
  });

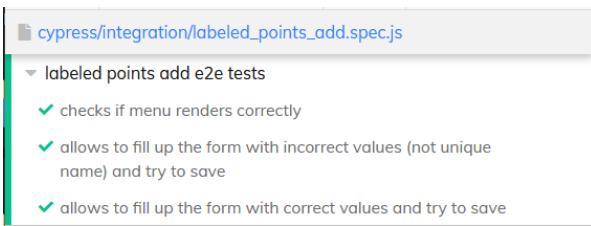
  it("allows to fill up the form with incorrect values (not unique name) and try to save", () => {
    cy.wait(1000);
    cy.get(':nth-child(1) > a > .LinkButton_linkText_2AiYn').click();
    cy.url().should("include", "dodaj");
    cy.wait(1000);
    cy.get('#pointNewName').clear();
    cy.get('#pointNewName').type('wełniNa');
    cy.wait(1000);
    cy.get('.Button_button_2iUVQ > p').click();
    cy.get('.AddLabeledPoint_info_3VS4x').should('exist');
    cy.get('.AddLabeledPoint_info_3VS4x').should('be.visible');
    cy.get('.AddLabeledPoint_info_3VS4x').should('have.text', 'Punkt o takiej nazwie już istnieje. Wybierz inną nazwę');
    cy.get('.LinkButton_linkText_2AiYn').should('exist');
    cy.get('.LinkButton_linkText_2AiYn').should('be.visible');
    cy.get('.LinkButton_linkText_2AiYn').should('have.text', 'Zakończ');
    cy.wait(1000);
    cy.get('.LinkButton_linkText_2AiYn').click();
  });
});
```

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

```

it("allows to fill up the form with correct values and try to save", () => {
  cy.wait(1000);
  cy.get(':nth-child(1) > a > .LinkButton_linkText__2AiYn').click();
  cy.url().should("include", "dodaj");
  cy.wait(1000);
  cy.get('#pointNewName').clear();
  cy.get('#pointNewName').type('Zielone Berdo');
  cy.get('#pointNewHeight').clear();
  cy.get('#pointNewHeight').type('989');
  cy.wait(1000);
  cy.get('.Button_button__2iUvQ > p').should('exist');
  cy.get('.Button_button__2iUvQ > p').should('be.visible');
  cy.get('.Button_button__2iUvQ > p').should('have.text', 'Dodaj punkt opisany');
  cy.get('.Button_button__2iUvQ > p').click();
  cy.get('.AddLabeledPoint_info__3VS4x').should('exist');
  cy.get('.AddLabeledPoint_info__3VS4x').should('be.visible');
  cy.get('.AddLabeledPoint_info__3VS4x').should('have.text', 'Punkt Zielone Berdo został pomyslnie dodany');
  cy.get('.LinkButton_linkText__2AiYn').should('exist');
  cy.get('.LinkButton_linkText__2AiYn').should('be.visible');
  cy.get('.LinkButton_linkText__2AiYn').should('have.text', 'Zakończ');
  cy.wait(1000);
  cy.get('.LinkButton_linkText__2AiYn').click();
});
});

```



“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

```

describe("labeled points update e2e tests", () => {
  beforeEach(() => {
    cy.visit("/punkt-opisany");
  });

  it("checks if menu renders correctly", () => {
    cy.get("#buttonContainer").should("exist");
    cy.get(':nth-child(1) > a > .LinkButton_linkText_2AiYn').should('exist');
    cy.get(':nth-child(1) > a > .LinkButton_linkText_2AiYn').should('be.visible');
    cy.get(':nth-child(1) > a > .LinkButton_linkText_2AiYn').should('have.text', 'Dodaj punkt opisany');
    cy.get(':nth-child(2) > a > .LinkButton_linkText_2AiYn').should('exist');
    cy.get(':nth-child(2) > a > .LinkButton_linkText_2AiYn').should('be.visible');
    cy.get(':nth-child(3) > a > .LinkButton_linkText_2AiYn').should('exist');
    cy.get(':nth-child(3) > a > .LinkButton_linkText_2AiYn').should('be.visible');
    cy.get(':nth-child(3) > a > .LinkButton_linkText_2AiYn').should('have.text', 'Usuń punkt opisany');
    cy.get(':nth-child(4) > a > .LinkButton_linkText_2AiYn').should('exist');
    cy.get(':nth-child(4) > a > .LinkButton_linkText_2AiYn').should('be.visible');
    cy.get(':nth-child(4) > a > .LinkButton_linkText_2AiYn').should('have.text', 'Szukaj punktów opisanych');
  });

  it("tries to edit point that doesn't exist", () => {
    cy.wait(1000);
    cy.get(':nth-child(2) > a > .LinkButton_linkText_2AiYn').click();
    cy.url().should("include", "edytuj");
    cy.wait(1000);
    cy.get('div > input').clear();
    cy.get('div > input').type('fdfhd');
    cy.wait(1000);
    cy.get('span').should('have.text', 'Nie znaleziono dopasowania');
    cy.get('.Button_button_2iUvQ').should('exist');
    cy.get('.Button_button_2iUvQ').should('be.visible');
    cy.get('.Button_button_2iUvQ').should('have.text', 'Edytuj punkt opisany');
    cy.get('.Button_button_2iUvQ').click();
    cy.wait(1000);
    cy.get('.EditLabeledPoint_info_3S-tF').should('exist');
    cy.get('.EditLabeledPoint_info_3S-tF').should('be.visible');
    cy.get('.EditLabeledPoint_info_3S-tF').should('have.text', 'Punkt opisany o nazwie: fdfhd nie istnieje');
    cy.get('.LinkButton_linkText_2AiYn').should('exist');
    cy.get('.LinkButton_linkText_2AiYn').should('be.visible');
    cy.get('.LinkButton_linkText_2AiYn').should('have.text', 'Zakończ');
    cy.wait(1000);
    cy.get('.LinkButton_linkText_2AiYn').click();
  });

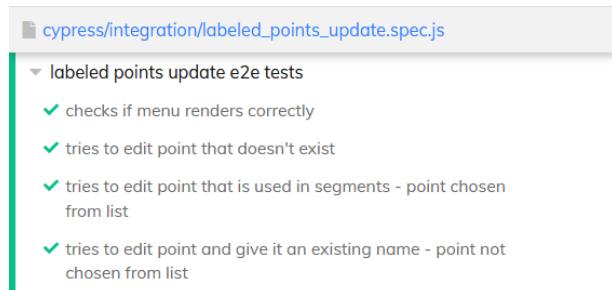
  it("tries to edit point that is used in segments - point chosen from list", () => {
    cy.wait(1000);
    cy.get(':nth-child(2) > a > .LinkButton_linkText_2AiYn').click();
    cy.url().should("include", "edytuj");
    cy.wait(1000);
    cy.get('div > input').clear();
    cy.get('div > input').type('wet');
    cy.get('#-\\31 389563850 > [data-user-value="true"]').click();
    cy.get('.Button_button_2iUvQ > p').should('exist');
    cy.get('.Button_button_2iUvQ > p').should('be.visible');
    cy.get('.Button_button_2iUvQ > p').should('have.text', 'Edytuj punkt opisany');
    cy.get('.Button_button_2iUvQ > p').click();
    cy.wait(1000);
    cy.get('.EditLabeledPoint_info_3S-tF').should('exist');
    cy.get('.EditLabeledPoint_info_3S-tF').should('be.visible');
    cy.get('.EditLabeledPoint_info_3S-tF').should('have.text', 'Punkt Wetlina jest już używany i nie można go edytować');
    cy.get('.LinkButton_linkText_2AiYn').should('exist');
    cy.get('.LinkButton_linkText_2AiYn').should('be.visible');
    cy.get('.LinkButton_linkText_2AiYn').should('have.text', 'Zakończ');
    cy.wait(1000);
    cy.get('.LinkButton_linkText_2AiYn').click();
  });
});

```

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

```

it("tries to edit point and give it an existing name - point not chosen from list", () => {
    cy.wait(1000);
    cy.get(':nth-child(2) > a > .LinkButton_linkText__2AiYn').click();
    cy.url().should("include", "edytuj");
    cy.wait(1000);
    cy.get('div > input').clear();
    cy.get('div > input').type('kozi wierch');
    cy.get('.Button_button__2iUvQ').should('be.visible');
    cy.get('.Button_button__2iUvQ').should('have.text', 'Edytuj punkt opisany');
    cy.wait(1000);
    cy.get('.Button_button__2iUvQ > p').click();
    cy.get('.LabeledPointForm_formWrapper__AaAYY > h2').should('have.text', 'Edycja punktu opisanego: kozi wierch');
    cy.get('#pointNewName').should('have.value', 'kozi wierch');
    cy.get('#pointNewHeight').should('have.value', '2291');
    cy.wait(1000);
    cy.get('#pointNewName').clear();
    cy.get('#pointNewName').type('wetlina');
    cy.wait(1000);
    cy.get('.Button_button__2iUvQ').click();
    cy.wait(1000);
    cy.get('.EditionManager_info__1RFmE').should('be.visible');
    cy.get('.EditionManager_info__1RFmE').should('have.text', 'Punkt o takiej nazwie już istnieje. Wybierz inną nazwę');
    cy.get('.LinkButton_linkText__2AiYn').should('be.visible');
    cy.get('.LinkButton_linkText__2AiYn').should('have.text', 'Zakończ');
    cy.wait(1000);
    cy.get('.LinkButton_linkText__2AiYn').click();
});
});
```



“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

2.2 Tworzenie trasy (Klaudia Błażyczek)

```

describe("labeled points add e2e tests", () => {
  beforeEach(() => {
    cy.visit("/tworzenie-trasy");
  });

  it("tour creation test (with non existing first point and non unique tour name at first attempts)", () => {
    cy.url().should("include", "tworzenie-trasy");
    cy.get('h3').should('be.visible');
    cy.get('h3').should('have.text', 'Stwórz trasę poprzez podanie punktów tworzących odcinki');
    cy.get('.TourCreationForm_pointsPresenter_Ok1jq').should('be.visible');
    cy.get('.TourCreationForm_pointsPresenter_Ok1jq').should('have.text', 'Punkty do GOT0');
    cy.get('.Button_button_2iUvQ > p').should('be.visible');
    cy.get('.Button_button_2iUvQ > p').should('have.text', 'Dodaj punkt startowy');
    cy.get('.Button_button_2iUvQ > p').click();
    cy.wait(1000);
    cy.get('.TourCreationForm_errorInfo_20Le4').should('be.visible');
    cy.get('.TourCreationForm_errorInfo_20Le4').should('have.text', 'Punkt opisany o nazwie: nie istnieje');
    cy.get('div > input').clear();
    cy.get('div > input').type('cvbvbcb');
    cy.get('.Button_button_2iUvQ').click();
    cy.wait(1000);
    cy.get('.TourCreationForm_errorInfo_20Le4').should('be.visible');
    cy.get('.TourCreationForm_errorInfo_20Le4').should('have.text', 'Punkt opisany o nazwie: cvbvbcb nie istnieje');
    cy.get('div > input').clear();
    cy.get('div > input').type('pol');
    cy.get('[data-user-value="true"]').click();
    cy.wait(1000);
    cy.get('.Button_button_2iUvQ > p').click();
    cy.get('h3').should('be.visible');
    cy.get('h3').should('have.text', 'Stwórz trasę poprzez podanie punktów tworzących odcinki');
    cy.get('.TourCreationForm_formWrapper_Dk6dc').should('be.visible');
    cy.get('.Button_button_2iUvQ').should('be.visible');
    cy.get('.Button_button_2iUvQ > p').should('have.text', 'Zatwierdź trasę');
    cy.get('.TourCreationForm_pointsPresenter_Ok1jq').should('be.visible');
    cy.get('.TourCreationForm_pointsPresenter_Ok1jq').should('have.text', 'Punkty do GOT0');
    cy.get(':nth-child(1) > p').should('be.visible');
    cy.get(':nth-child(1) > p').should('have.text', '+ Dodaj punkt');
    cy.get('.TourCreationForm_buttonsWrapper__VMfQ3 > :nth-child(2) > p').should('be.visible');
    cy.get('.TourCreationForm_buttonsWrapper__VMfQ3 > :nth-child(2) > p').should('have.text', '- Usuń ostatni punkt');
    cy.get('#button--listbox-input--57').click();
    cy.get('#option-6--listbox-input--57').click();
    cy.get(':nth-child(1) > p').click();
    cy.wait(1000);
    cy.get('#button--listbox-input--57').click();
    cy.get(':nth-child(1) > p').click();
    cy.wait(1000);
    cy.get('.TourCreationForm_buttonsWrapper__VMfQ3 > :nth-child(2)').click();
    cy.get('.TourCreationForm_buttonsWrapper__VMfQ3 > :nth-child(2) > p').click();
    cy.get('.TourCreationForm_buttonsWrapper__VMfQ3 > :nth-child(2) > p').click();
  });
});

```

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

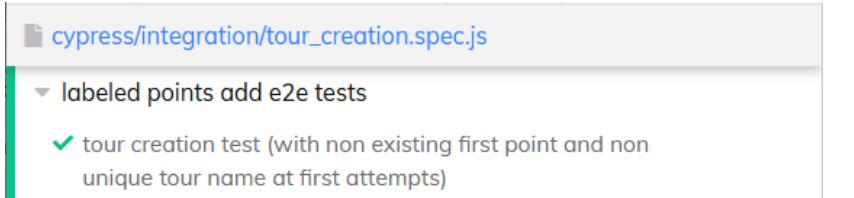
```

cy.get('div > input').clear();
cy.get('div > input').type('ho');
cy.get('[data-suggested-value="true"]').click();
cy.wait(1000);
cy.get('.Button_button_2iUvQ > p').click();
cy.get('.TourCreationForm_formWrapper_Dk6dc').should('be.visible');
cy.get('.Button_button_2iUvQ > p').click();
cy.get('.TourCreationForm_errorInfo_20Le4').should('be.visible');
cy.get('.TourCreationForm_errorInfo_20Le4').should('have.text', 'Nie wybrano punktu tworzącego odcinek');
cy.wait(1000);
cy.get('#button--listbox-input--159').click();
cy.get('#option-42-listbox-input--159').click();
cy.get('.TourCreationForm_buttonsWrapper_vMFQ3 > :nth-child(1)').click();
cy.get('.TourCreationForm_pointsPresenter_0k1jq > :nth-child(2)').should('be.visible');
cy.get('.TourCreationForm_pointsPresenter_0k1jq > :nth-child(2)').should('have.text', '13');
cy.wait(1000);
cy.get('.Button_button_2iUvQ > p').click();
cy.get('h3').should('be.visible');
cy.get('h3').should('have.text', 'Nadaj trasie nazwę');
cy.get('.Button_button_2iUvQ').should('be.visible');
cy.get('.Button_button_2iUvQ > p').should('have.text', 'Zatwierdź');
cy.get('.Button_button_2iUvQ').click();
cy.get('.TourCreationNameForm_errorInfo_zWjl2').should('be.visible');
cy.get('.TourCreationNameForm_errorInfo_zWjl2').should('have.text', 'Nie podano nazwy trasy');
cy.get('#tourName').click();
cy.wait(1000);
cy.get('#tourName').clear();
cy.get('#tourName').type('Bieszczady 2022');
cy.get('.Button_button_2iUvQ').click();
cy.get('.TourCreationNameForm_errorInfo_zWjl2').should('be.visible');
cy.get('.TourCreationNameForm_errorInfo_zWjl2').should('have.text', 'Trasa o wybranej nazwie już istnieje. Wybierz inną nazwę');
cy.wait(1000);
cy.get('#tourName').clear();
cy.get('#tourName').type('Bieszczady z rodziną 2022');
cy.get('.Button_button_2iUvQ').click();
cy.wait(1000);
cy.get('.TourCreation_resultInfo_2XMSM').should('be.visible');
cy.get('.TourCreation_resultInfo_2XMSM').should('have.text', 'Zapisano trasę: Bieszczady z rodziną 2022');
cy.get('.TourCreation_points_1LkhL').should('be.visible');
cy.get('.TourCreation_points_1LkhL').should('have.text', 'Liczba punktów do GOT: 13');
cy.get('tbody > tr > :nth-child(1)').should('be.visible');
cy.get('tbody > tr > :nth-child(1)').should('have.text', '1');
cy.get('tbody > tr > :nth-child(2)').should('be.visible');
cy.get('tbody > tr > :nth-child(2)').should('have.text', 'Hotel PTTK Ustrzyki Górné');
cy.get('tbody > tr > :nth-child(3)').should('be.visible');
cy.get('tbody > tr > :nth-child(3)').should('have.text', '-');
cy.get('tbody > tr > :nth-child(4)').should('be.visible');

    cy.get('tbody > tr > :nth-child(4)').should('have.text', 'Szeroki Wierch');
    cy.get('tbody > tr > :nth-child(5)').should('be.visible');
    cy.get('tbody > tr > :nth-child(5)').should('have.text', 'Bieszczady');
    cy.get('tbody > tr > :nth-child(6)').should('be.visible');
    cy.get('tbody > tr > :nth-child(6)').should('have.text', '13');
    cy.get('.LinkButton_linkText_2AiYn').should('be.visible');
    cy.get('.LinkButton_linkText_2AiYn').should('have.text', 'Zakończ');
    cy.wait(1000);
    cy.get('.LinkButton_linkText_2AiYn').click();
};

});

```



“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

2.3 Zarządzanie punktem własnym (Justyna Małuszyńska)

```

it("renders correctly", () => {
  cy.get("#buttonContainer").should("exist");
  cy.get(":nth-child(1) > a > .LinkButton_linkText_IcY_S").should("exist");
});

it("routes to add point form", () => {
  cy.get(":nth-child(1) > a > .LinkButton_linkText_IcY_S").click();
  cy.url().should("include", "dodaj");
});

it("allows to fill up the form with correct values and save", () => {
  cy.get(":nth-child(1) > a > .LinkButton_linkText_IcY_S").click();
  cy.url().should("include", "dodaj");
  cy.get("#pointNewName").clear();
  cy.get("#pointNewName").type("Nowy punkt własny");
  cy.get("#pointNewLatitude").clear();
  cy.get("#pointNewLatitude").type("45.6756");
  cy.get("#pointNewLongitude").clear();
  cy.get("#pointNewLongitude").type("43.4342");
  cy.get(".Button_button_2HkXA").click();
  cy.get('.AddOwnPoint_info_RMMLK').should('have.text', 'Punkt Nowy punkt własny został pomyślnie dodany');
  cy.get('.LinkButton_linkText_IcY_S').should('be.visible');
});

it("allows to fill up the form with incorrect values (not unique name) and save", () => {
  cy.get(":nth-child(1) > a > .LinkButton_linkText_IcY_S").click();
  cy.url().should("include", "dodaj");
  cy.get("#pointNewName").clear();
  cy.get("#pointNewName").type("Łysica");
  cy.get("#pointNewLatitude").clear();
  cy.get("#pointNewLatitude").type("45.6756");
  cy.get("#pointNewLongitude").clear();
  cy.get("#pointNewLongitude").type("43.4342");
  cy.get(".Button_button_2HkXA").click();
  cy.get('.AddOwnPoint_info_RMMLK').should('have.text', 'Punkt o takiej nazwie już istnieje. Wybierz inną nazwę');
  cy.get('.LinkButton_linkText_IcY_S').should('be.visible');
});

it("checks and print error messages while filling up the form", () => {
  cy.get(":nth-child(1) > a > .LinkButton_linkText_IcY_S").click();
  cy.url().should("include", "dodaj");
  cy.get("#pointNewName").clear();
  cy.get('#pointNewName').type('Jakaś nazwa');
  cy.get('.Button_button_2HkXA > p').click();
  cy.get('.OwnPointForm_errorInfo_1PB9a').should('have.text', 'Nie podano szerokości geograficznej punktu własnego');
  cy.get("#pointNewLatitude").clear();
  cy.get("#pointNewLatitude").type('12.3456');
  cy.get('.Button_button_2HkXA').click();
  cy.get('.OwnPointForm_errorInfo_1PB9a').should('have.text', 'Nie podano długości geograficznej punktu własnego');
  cy.get("#pointNewLongitude").clear();
  cy.get('#pointNewLongitude').type('45.5784');
  cy.get('.Button_button_2HkXA > p').click();
  cy.get('.AddOwnPoint_info_RMMLK').should('have.text', 'Punkt Jakaś nazwa został pomyślnie dodany');
  cy.get('.LinkButton_linkText_IcY_S').should('be.visible');
});

```

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

```

cypress/integration/own_points_add.spec.js
  ▼ renders own points page
    ✓ renders correctly
    ✓ routes to add point form
    ✓ allows to fill up the form with correct values and save
    ✓ allows to fill up the form with incorrect values (not unique name) and save
    ✓ checks and print error messages while filling up the form

```

2.4 Zgłaszanie odcinków trasy do weryfikacji (Justyna Małuszyńska)

```

it("renders correctly", () => {
  cy.get(".SearchForm_formWrapper__2KzEW > h2").should(
    "have.text",
    "Wybierz trasę do weryfikacji"
  );
  cy.get("div > input").should("exist");
  cy.get(".Button_button__2HkXA").should("exist");
  cy.get(".Button_button__2HkXA > p").should("have.text", "Szukaj trasy");
});

it("routes to tour segments list", () => {
  cy.get(".SearchForm_formWrapper__2KzEW > h2").should(
    "have.text",
    "Wybierz trasę do weryfikacji"
  );
  cy.get("div > input").should("exist");
  cy.get(".Button_button__2HkXA").should("exist");
  cy.get(".Button_button__2HkXA > p").should("have.text", "Szukaj trasy");
  cy.get("div > input").clear();
  cy.get("div > input").type("Moja");
  cy.get('#\\35 34561089 > [data-suggested-value="true"]').click();
  cy.get(".Button_button__2HkXA > p").click();
  cy.get(".EvidenceConfirmationSegmentsList_tableTitle__1fLfQ").should(
    "have.text",
    "Moja pierwsza trasa"
  );
  cy.get(".EvidenceConfirmationManager_info__L3l4i").should(
    "have.text",
    "Wybierz odcinki do weryfikacji"
  );
  cy.get("#segmentsList").should("exist");
  cy.get("tbody > :nth-child(1) > :nth-child(1)").should(
    "have.text",
    "Schronisko PTTK Hala Szrenicka"
  );
  cy.get("tbody > :nth-child(2) > :nth-child(1)").should(
    "have.text",
    "Wodospad Kamieńczyk"
  );
  cy.get("tbody > :nth-child(3) > :nth-child(1)").should(
    "have.text",
    "Schronisko PTTK Hala Szrenicka"
  );
});

```

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

```

    cy.get("tbody > :nth-child(4) > :nth-child(1)").should(
      "have.text",
      "Szrenica"
    );
    cy.get("tbody > :nth-child(5) > :nth-child(1)").should(
      "have.text",
      "Przełęcz Mokra"
    );
    cy.get("tbody > :nth-child(1) > :nth-child(2)").should(
      "have.text",
      "Wodospad Kamińczyk"
    );
    cy.get("tbody > :nth-child(2) > :nth-child(2)").should(
      "have.text",
      "Schronisko PTTK Hala Szrenicka"
    );
    cy.get("tbody > :nth-child(3) > :nth-child(2)").should(
      "have.text",
      "Szrenica"
    );
    cy.get("tbody > :nth-child(4) > :nth-child(2)").should(
      "have.text",
      "Przełęcz Mokra"
    );
    cy.get("tbody > :nth-child(5) > :nth-child(2)").should(
      "have.text",
      "Śnieżne Kotły"
    );
    cy.get(
      ".EvidenceConfirmationManager_wrapper__2Q04R > :nth-child(3)"
    ).should("exist");
  });
}

it("routes to data forms and let user fill that forms and pick tour segments", () => [
  cy.get(".SearchForm_formWrapper__2KzEW > h2").should(
    "have.text",
    "Wybierz trasę do weryfikacji"
  );
  cy.get("div > input").should("exist");
  cy.get(".Button_button__2HKXA").should("exist");
  cy.get(".Button_button__2HKXA > p").should("have.text", "Szukaj trasy");
  cy.get("div > input").clear();
  cy.get("div > input").type("Moja");
  cy.get('\u00a334561089 > [data-suggested-value="true"]').click();
  cy.get(".Button_button__2HKXA > p").click();
  cy.get("tbody > :nth-child(1) > :nth-child(1)").click();
  cy.get("tbody > :nth-child(2) > :nth-child(1)").click();
  cy.get(
    ".EvidenceConfirmationManager_wrapper__2Q04R > :nth-child(3)"
  ).click();
  cy.get(
    ":nth-child(2) > :nth-child(2) > .react-datepicker-wrapper > .react-datepicker_input-container > .EvidenceConfirmationDateForm_input__2UQDx"
  ).should("be.visible");
  cy.get(
    ":nth-child(2) > :nth-child(3) > .react-datepicker-wrapper > .react-datepicker_input-container > .EvidenceConfirmationDateForm_input__2UQDx"
  ).should("be.visible");
  cy.get(
    ":nth-child(3) > :nth-child(2) > .react-datepicker-wrapper > .react-datepicker_input-container > .EvidenceConfirmationDateForm_input__2UQDx"
  ).should("be.visible");
  cy.get(
    ":nth-child(3) > :nth-child(3) > .react-datepicker-wrapper > .react-datepicker_input-container > .EvidenceConfirmationDateForm_input__2UQDx"
  ).should("be.visible");
  cy.get(
    ":nth-child(2) > :nth-child(2) > .react-datepicker-wrapper > .react-datepicker_input-container > .EvidenceConfirmationDateForm_input__2UQDx"
  ).click();
]
)

```

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

```

cy.get(".react-datepicker__day--012").click();
cy.get(
  ".nth-child(2) > :nth-child(3) > .react-datepicker-wrapper > .react-datepicker__input-container > .EvidenceConfirmationDateForm_input__2UQDx"
).click();
cy.get(".react-datepicker__day--013").click();
cy.get(
  ".nth-child(3) > :nth-child(2) > .react-datepicker-wrapper > .react-datepicker__input-container > .EvidenceConfirmationDateForm_input__2UQDx"
).click();
cy.get(".react-datepicker__day--013").click();
cy.get(
  ".nth-child(3) > :nth-child(3) > .react-datepicker-wrapper > .react-datepicker__input-container > .EvidenceConfirmationDateForm_input__2UQDx"
).click();
cy.get(".react-datepicker__day--013").click();
cy.get(
  ".Button_button__2HkXA").click();
cy.get(":nth-child(2) > :nth-child(1) > p").should("be.visible");
cy.get(
  ".EvidenceConfirmationManager_wrapper__2Q04R > :nth-child(2) > :nth-child(2)"
).should("be.visible");
cy.get("tbody > :nth-child(2) > :nth-child(1)").should(
  "have.css",
  "background-color"
); //.and('eq', 'rgba(197, 221, 73, 0.25)');
cy.get("tbody > :nth-child(2) > :nth-child(2)").should(
  "have.css",
  "background-color"
); //.and('eq', 'rgba(197, 221, 73, 0.25)');
cy.get("tbody > :nth-child(2) > :nth-child(3)").should(
  "have.css",
  "background-color"
); //.and('eq', 'rgba(197, 221, 73, 0.25)');
cy.get("tbody > :nth-child(2) > :nth-child(4)").should(
  "have.css",
  "background-color"
); //.and('eq', 'rgba(197, 221, 73, 0.25)');
cy.get("tbody > :nth-child(1) > :nth-child(1)").should(
  "have.css",
  "background-color"
); //.and('eq', 'rgba(197, 221, 73, 0.25)');
cy.get("tbody > :nth-child(1) > :nth-child(2)").should(
  "have.css",
  "background-color"
); //.and('eq', 'rgba(197, 221, 73, 0.25)');
cy.get("tbody > :nth-child(1) > :nth-child(3)").should(
  "have.css",
  "background-color"
); //.and('eq', 'rgba(197, 221, 73, 0.25)');
cy.get("tbody > :nth-child(1) > :nth-child(4)").should(
  "have.css",
  "background-color"
); //.and('eq', 'rgba(197, 221, 73, 0.25)');
});
it("routes to guide form and shows error message", () => {
  cy.get(".SearchForm_formWrapper__2KzEW > h2").should(
    "have.text",
    "Wybierz trasę do weryfikacji"
  );
  cy.get("div > input").should("exist");
  cy.get(".Button_button__2HkXA").should("exist");
  cy.get(".Button_button__2HkXA > p").should("have.text", "Szukaj trasy");
  cy.get("div > input").clear();
  cy.get("div > input").type("Moja");
  cy.get("#\\35 34561089 > [data-suggested-value=true]").click();
  cy.get(".Button_button__2HkXA > p").click();
  cy.get("tbody > :nth-child(1) > :nth-child(1)").click();
  cy.get("tbody > :nth-child(2) > :nth-child(1)").click();
  cy.get(
    ".EvidenceConfirmationManager_wrapper__2Q04R > :nth-child(3)"
  ).click();
  cy.get(
    ":nth-child(2) > :nth-child(2) > .react-datepicker-wrapper > .react-datepicker__input-container > .EvidenceConfirmationDateForm_input__2UQDx"
  ).click();
  cy.get(".react-datepicker__day--012").click();
  cy.get(
    ":nth-child(2) > :nth-child(3) > .react-datepicker-wrapper > .react-datepicker__input-container > .EvidenceConfirmationDateForm_input__2UQDx"
  ).click();
  cy.get(".react-datepicker__day--013").click();
  cy.get(
    ":nth-child(3) > :nth-child(2) > .react-datepicker-wrapper > .react-datepicker__input-container > .EvidenceConfirmationDateForm_input__2UQDx"
  ).click();
  cy.get(".react-datepicker__day--013").click();
  cy.get(
    ":nth-child(3) > :nth-child(3) > .react-datepicker-wrapper > .react-datepicker__input-container > .EvidenceConfirmationDateForm_input__2UQDx"
  ).click();
  cy.get(".react-datepicker__day--013").click();
});

```

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

```

cy.get(".react-datepicker__day--013").click();
cy.get(".Button_button_2HkXA").click();
cy.get(":nth-child(2) > :nth-child(2) > p").click();
cy.get(".EvidenceConfirmationEvidenceForm_formWrapper_13cPt > h2").should(
  "have.text",
  "Wpisz dane przewodnika odbywającego z Tobą trasę"
);
cy.get("#evidence").should("have.id", "evidence");
cy.get("#evidence").clear();
cy.get("#evidence").type("654321");
cy.get(".Button_button_2HkXA").click();
cy.get(".EvidenceConfirmationEvidenceForm_errorInfo_3rP08").should(
  "be.visible"
);
cy.get(".EvidenceConfirmationEvidenceForm_errorInfo_3rP08").should(
  "have.text",
  "Przewodnik o podanym numerze legitymacji nie istnieje"
);
});

```

 cypress/integration/evidence_confirmation.spec.js

- ▼ renders own points page
 - ✓ renders correctly
 - ✓ routes to tour segments list
 - ✓ routes to data forms and let user fill that forms and pick tour segments
 - ✓ routes to guide form and shows error message

3. Testy jednostkowe

Testy jednostkowe dla dodawania punktu opisanego (Klaudia Błażyczek)

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

```

5   # common case
6 | # testing /labeled-points POST endpoint for new point with height
7 | # this endpoint should return message and status code 200
8 def test_add_point_with_correct_name_and_given_height(client):
9     new_point = {
10         "name": "Punkt testowy 1",
11         "height": 1245
12     }
13
14     response = client.post(
15         "/labeled-points",
16         data = json.dumps(new_point),
17         headers = {"Content-Type": "application/json"},
18     )
19
20     message = json.loads(response.data.decode('utf-8')).get("message")
21
22     assert response.status_code == 200
23     assert message == "Punkt został pomyślnie dodany"
24
25     """
26     # common case
27     # testing /labeled-points POST endpoint for new point with no height
28     # this endpoint should return message and status code 200
29     def test_add_point_with_correct_name_and_no_height(client):
30         new_point = {
31             "name": "Punkt testowy 2",
32
PROBLEMS 67 OUTPUT DEBUG CONSOLE TERMINAL
..\.venv\lib\site-packages\f flask_autodoc\autodoc.py:55           \.venv\lib\site-packages\f flask_autodoc\autodoc.py:55: DeprecationWarning
ng: 'evalcontextfilter' is renamed to 'pass_eval_context', the old name will be removed in Jinja 3.1.
def nl2br(eval_ctx, value):
-- Docs: https://docs.pytest.org/en/stable/warnings.html
===== PASSES =====
===== 1 passed, 1 warning in 1.75s =====

```

Powyższy test jednostkowy miał na celu przetestować przypadek, w którym członek komisji chciałby dodać nieistniejący punkt opisany do bazy, podając przy tym poprawne dane (dodatnią wartość wysokości oraz unikalną nazwę). W tym celu tworzony jest więc nowy punkt, następnie wysyłane jest zapytanie POST, a odpowiedź zapytania porównuje się z oczekiwanymi wartościami, czyli statusem odpowiedzi 200 oraz informacją zwrotną. Endpoint /labeled-points dla akcji POST wywołuje pod spodem metodę add_labeled_point z pakietu services i to właśnie poprawność działania tej metody jest testowana.

```

46      # edge case
47  ● 48      # testing /labeled-points POST endpoint for new point with non unique name and invalid height
49      # non unique name will be caught first
50      # this endpoint should return message with error and status code 400
51  ▼ def test_add_point_with_incorrect_name_and_incorrect_height(client):
52      new_point = {
53          "name": "weTliNa", # point with this name exist in DB. Also shows that it is not case sensitive
54          "height": -456
55      }
56
57  ▼ response = client.post(
58      "/labeled-points",
59      data = json.dumps(new_point),
60      headers = {"Content-Type": "application/json"},
61  )
62
63      message = json.loads(response.data.decode('utf-8')).get("message")
64
65      assert response.status_code == 400
66      assert message == "Punkt o takiej nazwie już istnieje. Wybierz inną nazwę"
67
PROBLEMS 107 OUTPUT DEBUG CONSOLE TERMINAL
===== test session starts =====
platform win32 -- Python 3.9.7, pytest-6.2.5, py-1.11.0, pluggy-1.0.0
rootdir: C:\Users\Kacper\PycharmProjects\turysta_z_odznaką
collected 1 item

test_labeled_pointsAPI_add.py . [100%]

===== warnings summary =====
..\venv\lib\site-packages\flask_autodoc\autodoc.py:55
    ..\venv\lib\site-packages\flask_autodoc\autodoc.py:55: DeprecationWarning: 'evalcontextfilter' is renamed to 'pass_eval_context', the old name will be removed in Jinja 3.1.
        def nl2br(eval_ctx, value):

-- Docs: https://docs.pytest.org/en/stable/warnings.html
===== PASSES =====
===== 1 passed, 1 warning in 1.91s =====

```

Powyższy test jednostkowy miał na celu przetestować przypadek, w którym członek komisji chciałby dodać punkt opisany do bazy, podając przy tym niepoprawne dane (ujemną wartość wysokości oraz nieunikalną nazwę). W tym celu tworzony jest więc nowy punkt z błędnymi danymi, następnie wysyłane jest zapytanie POST, a odpowiedź zapytania porównuje się z oczekiwanyymi wartościami, czyli statusem odpowiedzi 400 oraz informacją zwotną. W przypadku, gdy zarówno nazwa jak i wysokość są niepoprawne, zwracany błąd będzie dotyczył nazwy, gdyż unikalność nazwy jest sprawdzana przed poprawnością wysokości. Warto zaznaczyć, że punkt o nazwie Wetlina istnieje w bazie danych. Endpoint /labeled-points dla akcji POST wywołuje pod spodem metodę add_labeled_point z pakietu services i to właśnie poprawność działania tej metody jest testowana.

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

```

backend > tests > 📁 test_labeled_pointsAPI_add.py > ...
90  # edge case
91  # testing /labeled-points POST endpoint for new point with incorrect height (< 0)
92  # this endpoint should return message with error and status code 400
93 def test_add_point_with_incorrect_height(client):
94     new_point = {
95         "name": "Punkt testowy 3",
96         "height": -23
97     }
98
99     response = client.post(
100         "/labeled-points",
101         data = json.dumps(new_point),
102         headers = {"Content-Type": "application/json"},
103     )
104
105     message = json.loads(response.data.decode('utf-8')).get("message")
106
107     assert response.status_code == 400
108     assert message == "Niepoprawna wartość wysokości"
109
PROBLEMS 107  OUTPUT  DEBUG CONSOLE  TERMINAL
rootdir: C:\Users\Kuba\PycharmProjects\turysta_z_odznaką
collected 1 item
test_labeled_pointsAPI_add.py .
===== warnings summary =====
..\venv\lib\site-packages\flask_autodoc\autodoc.py:55: DeprecationWarning: 'evalcontextfilter' is renamed to 'pass_eval_context', the old name will be removed in Jinja 3.1.
    def nl2br(eval_ctx, value):
-- Docs: https://docs.pytest.org/en/stable/warnings.html
===== PASSES =====
===== 1 passed, 1 warning in 1.93s =====
[100%]

```

Powyższy test jednostkowy miał na celu przetestować przypadek, w którym członek komisji chciałby dodać punkt opisany do bazy, podając przy tym niepoprawną wysokość punktu. W tym celu tworzony jest więc nowy punkt z błędnymi danymi, następnie wysyłane jest zapytanie POST, a odpowiedź zapytania porównuje się z oczekiwany wartościami, czyli statusem odpowiedzi 400 oraz informacją zwrotną. Endpoint /labeled-points dla akcji POST wywołuje pod spodem metodę add_labeled_point z pakietu services i to właśnie poprawność działania tej metody jest testowana.

W screenach testów oraz ich wyników pozwoliłam sobie zamazać ścieżki na moim dysku.

Kod źródłowy do testów jednostkowych dodawania punktu opisanego znajduje się na ścieżce: backend\tests\test_labeled_pointsAPI_add.py

W pliku tym znajdują się jeszcze: test dla akcji dodawania poprawnego punktu bez wysokości oraz test dla akcji dodawania poprawnego z nieunikalną nazwą.

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

Testy jednostkowe dla edycji punktu opisanego (Klaudia Błażyczek)

```

4
5     # common case
6     # testing /labeled-points/<int:id> PUT endpoint for point - correct new data
7     # this endpoint should return message and status code 200
8     def test_update_point_correct_new_data(client):
9         new_point_data = {
10            "name": "Szkalna góra",
11            "height": 1234
12        }
13
14        response = client.put(
15            "/labeled-points/54", # point with id 54 is not used in segments yet
16            data = json.dumps(new_point_data),
17            headers = {"Content-Type": "application/json"},
18        )
19
20        message = json.loads(response.data.decode('utf-8')).get("message")
21
22        assert response.status_code == 200
23        assert message == "Punkt został pomyślnie edytowany"
24

```

PROBLEMS 127 OUTPUT DEBUG CONSOLE TERMINAL

rootdir: [REDACTED]
collected 1 item

test_labeled_pointsAPI_update.py . [100%]

----- warnings summary -----
..\\venv\\lib\\site-packages\\flask_autodoc\\autodoc.py:55 ..\\venv\\lib\\site-packages\\flask_autodoc\\autodoc.py:55: DeprecationWarning: 'evalcontextfilter' is renamed to 'pass_eval_context', the old name will be removed in Jinja 3.1.
def nl2br(eval_ctx, value):
-- Docs: <https://docs.pytest.org/en/stable/warnings.html> ----- PASSES -----
----- 1 passed, 1 warning in 1.81s -----

Powyższy test jednostkowy miał na celu przetestować przypadek, w którym członek komisji chciałby edytować punkt opisany, podając przy tym poprawne dane (dodatkową wartość wysokości i unikalną nazwę dla punktu, oraz punkt, którego dane są zmieniane, nie jest używany w odcinkach). W tym celu tworzony jest więc nowy punkt, który niejako zastąpi stary punkt, następnie wysyłane jest zapytanie PUT, a odpowiedź zapytania porównuje się z oczekiwanyymi wartościami, czyli statusem odpowiedzi 200 oraz informacją zwrotną. Warto dodać, że w bazie nie ma punktu o nazwie Szklana góra, a punkt o id 54 nie jest używany w odcinkach. Endpoint /labeled-points dla akcji PUT wywołuje pod spodem metodę update_labeled_point z pakietu services i to właśnie poprawność działania tej metody jest testowana.

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

```

55
69 # edge case
70 # testing /labeled-points/<int:id> PUT endpoint for point where new name is already taken by another point and height is invalid
71 # non unique name will be caught first
72 # this endpoint should return message with error and status code 400
73 def test_update_point_with_existing_name_and_incorrect_height(client):
74     new_point_data = {
75         "name": "Siklawa",
76         "height": -456
77     }
78
79     response = client.put(
80         "/labeled-points/1",
81         data = json.dumps(new_point_data),
82         headers = {"Content-Type": "application/json"},
83     )
84
85     message = json.loads(response.data.decode('utf-8')).get("message")
86
87     assert response.status_code == 400
88     assert message == "Punkt o takiej nazwie już istnieje. Wybierz inną nazwę"
89

```

PROBLEMS 127 OUTPUT DEBUG CONSOLE TERMINAL

```

rootdir: C:\Users\...\Desktop\...
collected 1 item

test_labeled_pointsAPI_update.py . [100%]

===== warnings summary =====
..\venv\lib\site-packages\flask_autodoc\autodoc.py:55 ..... \venv\lib\site-packages\flask_autodoc\autodoc.py:55: DeprecationWarning
ng: 'evalcontextfilter' is renamed to 'pass_eval_context', the old name will be removed in Jinja 3.1.
def nl2br(eval_ctx, value):
-- Docs: https://docs.pytest.org/en/stable/warnings.html
===== PASSES =====
===== 1 passed, 1 warning in 1.71s =====

```

Powyższy test jednostkowy miał na celu przetestować przypadek, w którym członek komisji chciałby edytować punkt opisany, podając przy tym niepoprawne dane (ujemną wartość wysokości i nieunikalną nazwę dla punktu). W tym celu tworzony jest więc nowy punkt, który niejako zastąpi stary punkt, następnie wysyłane jest zapytanie PUT, a odpowiedź zapytania porównuje się z oczekiwanyymi wartościami, czyli statusem odpowiedzi 400 oraz informacją zwrotną. Warto dodać, że w bazie jest już punktu o nazwie Siklawa, a punkt o id 1 nie jest używany w odcinkach. Endpoint /labeled-points dla akcji PUT wywołuje pod spodem metodę update_labeled_point z pakietu services i to właśnie poprawność działania tej metody jest testowana.

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

```

112 # edge case
113 # testing /labeled-points/<int:id> PUT endpoint for point where new name is already taken by another point
114 # this endpoint should return message with error and status code 400
115 def test_update_point_with_existing_name(client):
116     new_point_data = {
117         "name": "wetlina",
118         "height": None
119     }
120
121     response = client.put(
122         "/labeled-points/1",
123         data = json.dumps(new_point_data),
124         headers = {"Content-Type": "application/json"},
125     )
126
127     message = json.loads(response.data.decode('utf-8')).get("message")
128
129     assert response.status_code == 400
130     assert message == "Punkt o takiej nazwie już istnieje. Wybierz inną nazwę"
131

```

PROBLEMS 127 OUTPUT DEBUG CONSOLE TERMINAL

```

platform win32 -- Python 3.9.7, pytest-6.2.5, py-1.11.0, pluggy-1.0.0
rootdir: C:\Users\Kacper\PycharmProjects\turysta_z_odznaką\services\labeled-points\tests
collected 1 item

test_labeled_pointsAPI_update.py . [100%]

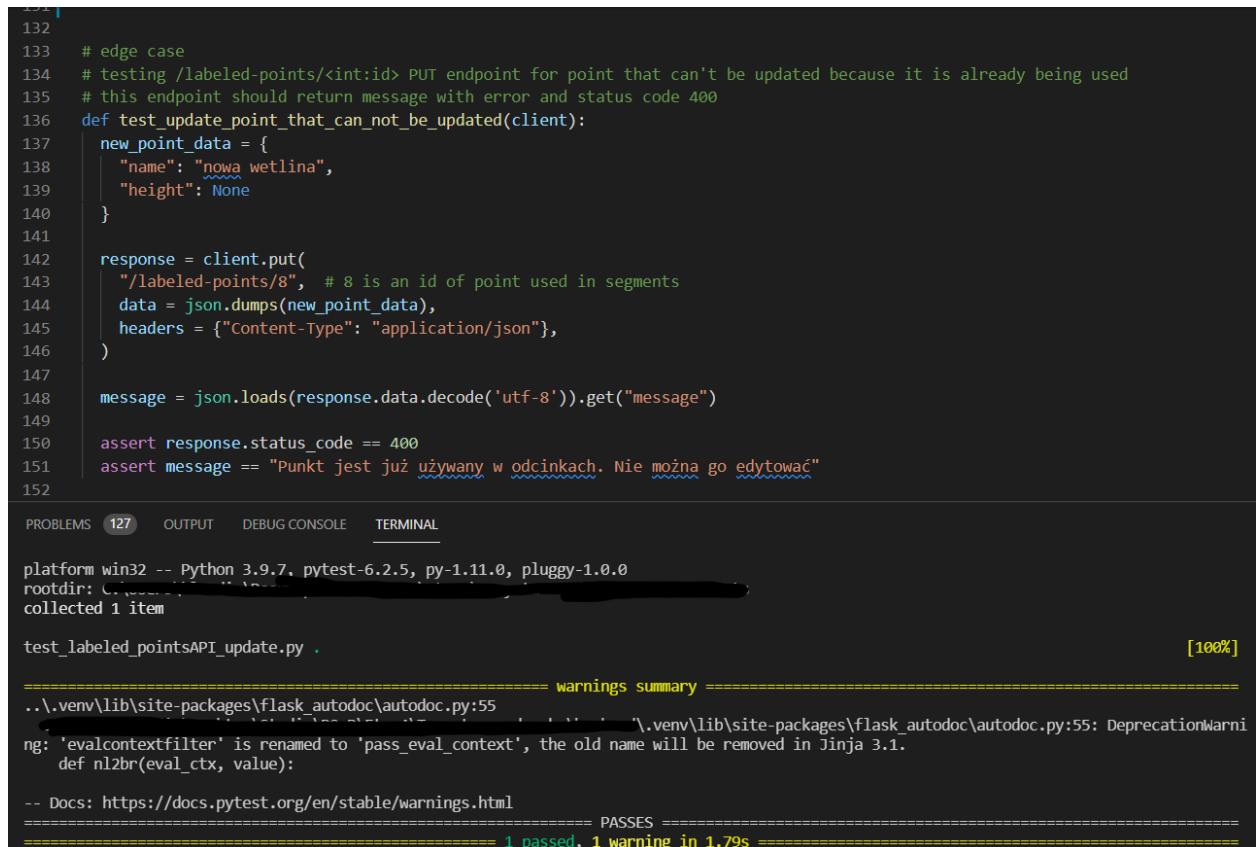
===== warnings summary =====
..\\venv\\lib\\site-packages\\flask_autodoc\\autodoc.py:55: DeprecationWarning: 'evalcontextfilter' is renamed to 'pass_eval_context', the old name will be removed in Jinja 3.1.
def nl2br(eval_ctx, value):

-- Docs: https://docs.pytest.org/en/stable/warnings.html
===== PASSES =====
===== 1 passed, 1 warning in 1.78s =====

```

Powyższy test jednostkowy miał na celu przetestować przypadek, w którym członek komisji chciałby edytować punkt opisany, podając przy tym niepoprawne dane (nieunikalną nazwę dla punktu). W tym celu tworzony jest więc nowy punkt, który niejako zastąpi stary punkt, następnie wysyłane jest zapytanie PUT, a odpowiedź zapytania porównuje się z oczekiwanyymi wartościami, czyli statusem odpowiedzi 400 oraz informacją zwrotną. Warto dodać, że w bazie jest już punktu o nazwie Wetlina, a punkt o id 1 nie jest używany w odcinkach. Endpoint /labeled-points dla akcji PUT wywołuje pod spodem metodę update_labeled_point z pakietu services i to właśnie poprawność działania tej metody jest testowana.

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>



```

132
133 # edge case
134 # testing /labeled-points/<int:id> PUT endpoint for point that can't be updated because it is already being used
135 # this endpoint should return message with error and status code 400
136 def test_update_point_that_can_not_be_updated(client):
137     new_point_data = {
138         "name": "nowa wetlina",
139         "height": None
140     }
141
142     response = client.put(
143         "/labeled-points/8", # 8 is an id of point used in segments
144         data = json.dumps(new_point_data),
145         headers = {"Content-Type": "application/json"},
146     )
147
148     message = json.loads(response.data.decode('utf-8')).get("message")
149
150     assert response.status_code == 400
151     assert message == "Punkt jest już używany w odcinkach. Nie można go edytować"
152

```

PROBLEMS 127 OUTPUT DEBUG CONSOLE TERMINAL

```

platform win32 -- Python 3.9.7, pytest-6.2.5, py-1.11.0, pluggy-1.0.0
rootdir: C:\Users\Kacper\PycharmProjects\turysta_z_odznaką\labeled_points
collected 1 item

test_labeled_pointsAPI_update.py . [100%]

===== warnings summary =====
..\venv\lib\site-packages\flask_autodoc\autodoc.py:55: DeprecationWarning: 'evalcontextfilter' is renamed to 'pass_eval_context', the old name will be removed in Jinja 3.1.
    def nl2br(eval_ctx, value):
-- Docs: https://docs.pytest.org/en/stable/warnings.html
===== PASSES =====
===== 1 passed, 1 warning in 1.79s =====

```

Powyższy test jednostkowy miał na celu przetestować przypadek, w którym członek komisji chciałby edytować punkt opisany, który jest już używany w odcinkach, podając przy tym poprawne nowe dane (unikalną nazwę dla punktu i poprawną wartość wysokości). W tym celu tworzony jest więc nowy punkt, który niejako zastąpi stary punkt, następnie wysyłane jest zapytanie PUT, a odpowiedź zapytania porównuje się z oczekiwany wartościami, czyli statusem odpowiedzi 400 oraz informacją zwrotną. Warto dodać, punkt o id 8 jest używany w odcinkach. Endpoint /labeled-points dla akcji PUT wywołuje pod spodem metodę update_labeled_point z pakietu services i to właśnie poprawność działania tej metody jest testowana.

W screenach testów oraz ich wyników pozwoliłam sobie zamazać ścieżki na moim dysku.

Kod źródłowy do testów jednostkowych edytowania punktu opisanego znajduje się na ścieżce: backend\tests\test_labeled_pointsAPI_update.py

W pliku tym znajdują się jeszcze:

- test dla edycji punktu nieużywanego w odcinkach na punkt o tej samej nazwie, ale nowej wysokości,
- test dla edycji punktu używanego w odcinkach i próba zmiany jego nazwy na tą samą
- test dla edycji punktu nieużywanego w odcinkach na punkt z unikalną nazwą, ale ujemną wysokością
- test dla edycji nieistniejącego punktu

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

Testy jednostkowe dla wyszukiwania punktów opisanych zawierających podaną frazę (Klaudia Błażyczek)

```

34
35     # common case
36     # testing /labeled-points/like/<string:like> GET endpoint for phrase that matches to many points
37     # this endpoint should return list of matching points and status code 200
38     def test_like_with_matched_to_many_points_string(client):
39         ids_of_all_labeled_points = [1, 4, 12, 13, 22, 46, 48, ]
40         unmatched_phrase = "ko"
41
42         response = client.get("/labeled-points/like/" + unmatched_phrase)
43         labeled_points = json.loads(response.data.decode('utf-8'))
44         labeled_points_ids = list(map(lambda point: point['id'], labeled_points))
45
46         assert response.status_code == 200
47         assert ids_of_all_labeled_points == labeled_points_ids
48

```

PROBLEMS 127 OUTPUT DEBUG CONSOLE TERMINAL

(.venv) PS C:\Users\Klaudia\PycharmProjects\turysta_z_odznaką> pytest -rP .\test_labeled_pointsAPI_like.py

===== test session starts =====

platform win32 -- Python 3.9.7, pytest-6.2.5, py-1.11.0, pluggy-1.0.0

rootdir: C:\Users\Klaudia\PycharmProjects\turysta_z_odznaką

collected 1 item

test_labeled_pointsAPI_like.py . [100%]

===== warnings summary =====

..\venv\lib\site-packages\flask_autodoc\autodoc.py:55: DeprecationWarning: 'evalcontextfilter' is renamed to 'pass_eval_context', the old name will be removed in Jinja 3.1.

def nl2br(eval_ctx, value):

[Open file in editor (ctrl + click)] st.org/en/stable/warnings.html

PASSES =====

1 passed, 1 warning in 1.66s =====

Powyższy test jednostkowy miał na celu przetestować przypadek, w którym turysta chciałby wyszukać punkty opisane, zawierające frazę „ko”. W tym celu zapisuję do tablicy id wszystkich punktów opisanych zawierających daną frazę, a następnie sprawdzam z id punktów opisanych zwroconych przez zapytanie GET. Sprawdzany jest też status odpowiedzi, który powinien wynieść 200. Endpoint /labeled-points/like/<string:like> dla akcji GET wywołuje pod spodem metodę get_labeled_points_like z pakietu services i to właśnie poprawność działania tej metody jest testowana.

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

```
50 # edge case
51 # testing /labeled-points/like/<string:like> GET endpoint for empty string
52 # this endpoint should return all labeled points and status code 200
53 def test_like_with_empty_string(client):
54     ids_of_all_labeled_points = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,42,43,44,45,46,47,48,49,50,51,52,53,54]
55
56     response = client.get("/labeled-points/like/")
57     labeled_points = json.loads(response.data.decode('utf-8'))
58     labeled_points_ids = list(map(lambda point: point['id'], labeled_points))
59
60     assert response.status_code == 200
61     assert ids_of_all_labeled_points == labeled_points_ids
62
63
PROBLEMS 80 OUTPUT DEBUG CONSOLE TERMINAL
(.venv) PS C:\Users\...> pytest -n 1 -P .\test_labeled_pointsAPI_like.py
platform win32 -- Python 3.9.7, pytest-6.2.5, py-1.11.0, pluggy-1.0.0
rootdir: C:\Users\...
collected 1 item

test_labeled_pointsAPI_like.py . [100%]

----- warnings summary -----
..\venv\lib\site-packages\flask_autodoc\autodoc.py:55: DeprecationWarning
ng: 'evalcontextfilter' is renamed to 'pass_eval_context', the old name will be removed in Jinja 3.1.
    def nl2br(eval_ctx, value):

-- Docs: https://docs.pytest.org/en/stable/warnings.html
----- PASSES -----
----- 1 passed, 1 warning in 1.92s -----
```

Powyższy test jednostkowy miał na celu przetestować przypadek, w którym turysta chciałby wyszukać punkty opisane, zawierające frazę „”, czyli pusty ciąg znaków. W tym celu zapisuję do tablicy id wszystkich punktów opisanych z bazy danych, a następnie sprawdzam z id punktów opisanych zwróconych przez zapytanie GET, gdyż metoda powinna zwrócić wszystkie punkty opisane. Sprawdzany jest też status odpowiedzi, który powinien wynieść 200. Endpoint /labeled-points/like/ dla akcji GET wywołuje pod spodem metodę get_labeled_points z pakietu services i to właśnie poprawność działania tej metody jest testowana.

W screenach testów oraz ich wyników pozwoliłam sobie zamazać ścieżki na moim dysku.

Kod źródłowy do testów jednostkowych wyszukiwania punktów opisanych zawierających podaną frazę znajduje się na ścieżce: backend\tests\test_labeled_pointsAPI_like.py

W pliku tym znajdują się jeszcze:

- test dla frazy, której nie zawiera żaden punkt opisany
 - kilka testów podobnych do pierwszego testu

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

Testy jednostkowe dla tworzenia trasy (Klaudia Błażyczek)

```
# common case
# testing /tour-creation/tour POST endpoint for creating tour with points that were once closed in the past
# this endpoint should return created labeled segments and status code 200
def test_create_tour_with_closed_segments_in_past(client):
    new_tour_data = {
        "name": "Bieszczady z rodziną 2022",
        "points": 21,
        "segments": [
            {
                "closed_segments": [],
                "color": "niebieski",
                "end_point_id": 43,
                "id": 6,
                "is_bidirectional": True,
                "liquidated_segment": None,
                "mountain_group_id": 1,
                "points": 12,
                "start_point_id": 42,
                "through": None
            },
            {
                "closed_segments": [
                    {
                        "closureDate": "2020-11-07",
                        "id": 8,
                        "openingDate": "2021-03-03"
                    }
                ],
                "color": "niebieski",
                "end_point_id": 12,
                "id": 11,
                "is_bidirectional": True,
                "liquidated_segment": None,
                "mountain_group_id": 1,
                "points": 6,
                "start_point_id": 43,
                "through": None
            },
        ]
    }
```

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

```

    {
        "closed_segments": [],
        "color": "brak",
        "end_point_id": 47,
        "id": 20,
        "is_bidirectional": True,
        "liquidated_segment": None,
        "mountain_group_id": 1,
        "points": 3,
        "start_point_id": 12,
        "through": None
    }
}

# confirm that tour name unique
response_for_check_name = client.post(
    "/tour-creation/check-name",
    data = json.dumps({"name": new_tour_data["name"]}),
    headers = {"Content-Type": "application/json"},
)

if_name_unique = json.loads(response_for_check_name.data.decode('utf-8'))

response_for_tour_creation = client.post(
    "/tour-creation/tour",
    data = json.dumps(new_tour_data),
    headers = {"Content-Type": "application/json"},
)

tour_segments = json.loads(response_for_tour_creation.data.decode('utf-8'))

# confirm GOT points were calculated correctly
GOT_points = 0

for segment in tour_segments:
    GOT_points += segment["points"]

```

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

```

# confirm segments contiguous and not closed / liquidated in points
if_contiguous_not_closed_not_liquidated = True

for i in range(0, len(tour_segments) - 1):
    # check if contiguous
    if (tour_segments[i]["labeled_segment"]["end_point"]["id"] != tour_segments[i + 1]["labeled_segment"]["start_point"]["id"]):
        if_contiguous_not_closed_not_liquidated = False

    # check if not liquidated
    if (tour_segments[i]["labeled_segment"]["liquidated_segment"] != None):
        if_contiguous_not_closed_not_liquidated = False

    # check if currently not closed
    if (tour_segments[i]["labeled_segment"]["closed_segments"] != None):
        for closedSegment in tour_segments[i]["labeled_segment"]["closed_segments"]:
            todays_date = datetime.today().date()
            closure_date = datetime.strptime(closedSegment["closureDate"], '%Y-%m-%d').date()

            if (closedSegment["openingDate"] == None):
                if (todays_date >= closure_date):
                    if_contiguous_not_closed_not_liquidated = False
            else:
                opening_date = datetime.strptime(closedSegment["openingDate"], '%Y-%m-%d').date()

                if (todays_date >= closure_date and todays_date <= opening_date):
                    if_contiguous_not_closed_not_liquidated = False

assert response_for_check_name.status_code == 200
assert if_name_unique == True

assert response_for_tour_creation.status_code == 200
assert GOT_points == new_tour_data["points"]
assert if_contiguous_not_closed_not_liquidated == True

```

collected 1 item

test_tour_creationAPI_tour_creation.py .

----- Warnings summary -----

..\venv\lib\site-packages\flask_autodoc\autodoc.py:55 ..\venv\lib\site-packages\flask_autodoc\autodoc.py:55: DeprecationWarning: 'evalcontextfilter' is renamed to 'pass_eval_context', the old name will be removed in Jinja 3.1.
def nI2Br(eval_ctx, value):

-- Docs: <https://docs.pytest.org/en/stable/warnings.html>

----- 1 passed, 1 warning in 1.88s -----

Ze względu na to, jak dla przypadku użycia został zaimplementowany interfejs użytkownika, niemożliwym jest stworzenie trasy z nieunikalną nazwą lub składający się z odcinków: nieciągłych w punktach, nieczynnych, zlikwidowanych. Zapewnia to wyżej wymieniony interfejs użytkownika. Ponowna weryfikacja nie występuje więc na backendzie (w porozumieniu z prowadzącym). Z tego względu test jednostkowy sprawdza jedynie, czy faktycznie powyższe założenia są spełnione.

Na początku tworzona jest trasa (JSON z trasą) zawierająca nazwę, sumę punktów GOT oraz odcinki opisane wchodzące w skład trasy. Kolejnym krokiem jest sprawdzenia unikalności nazwy trasy dla danego turysty (do tego stworzony został osobny endpoint, dla którego również napisane są testy jednostkowe). Następnie wysyłane jest zapytanie POST, które pośrednio wywołuje metodę add_tour_and_tour_segments z pakietu services (to właśnie poprawność działania tej metody jest testowana). Odpowiedź z zapytania POST jest zapisywana do zmiennej. Kolejnym krokiem jest sprawdzenie, czy punktacja GOT w odpowiedzi została poprawnie podliczona. Następnie sprawdzane jest, czy punkt końcowy odcinka trasy, jest punktem początkowym kolejnego odcinka trasy. Sprawdzane jest też, czy odcinek trasy nie jest zlikwidowany, lub czasowo nieczynny.

W screenach testów oraz ich wyników pozwoliłam sobie zamazać ścieżki na moim dysku.

W pliku testów tworzenia trasy (backend\tests\test_tour_creationAPI_tour_creation.py) znajduje się drugi test, działający na podobnej zasadzie.

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

Testy jednostkowe dla sprawdzania unikalności nazwy trasy dla turysty (Klaudia Błażyczek)

```

5   # common case
6   # testing /tour-creation/check-name POST endpoint for tour name - tourist doesn't have tour with the name
7   # this endpoint should return message and status code 200
8   def test_check_name_that_not_exists(client):
9       tour_name = {
10           "name": "Bieszczadzka przygoda"
11       }
12
13       response = client.post(
14           "/tour-creation/check-name",
15           data = json.dumps(tour_name),
16           headers = {"Content-Type": "application/json"},
17       )
18
19       if_name_unique = json.loads(response.data.decode('utf-8'))
20
21       assert response.status_code == 200
22       assert if_name_unique == True
23
24
25   # edge case
26   # testing /tour-creation/check-name POST endpoint for tour name - tourist already has tour with the name
27   # this endpoint should return message and status code 400
28   def test_check_name_that_exists(client):
29       tour_name = {
30           "name": "Bieszczady 2022"
31       }
32
33       response = client.post(
34           "/tour-creation/check-name",
35           data = json.dumps(tour_name),
36           headers = {"Content-Type": "application/json"},
37       )
38
39       message = json.loads(response.data.decode('utf-8')).get("message")
40
41       assert response.status_code == 400
42       assert message == "Trasa o wybranej nazwie już istnieje. Wybierz inną nazwę"

```

PROBLEMS 50 OUTPUT DEBUG CONSOLE TERMINAL

```

..\.venv\lib\site-packages\flask_autodoc\autodoc.py:55 ..\.venv\lib\site-packages\flask_autodoc\autodoc.py:55: DeprecationWarning: the old name will be removed in Jinja 3.1.
def nl2br(eval_ctx, value):

-- Docs: https://docs.pytest.org/en/stable/warnings.html
===== PASSES =====
===== 2 passed, 1 warning in 1.76s =====

```

Powyższe testy jednostkowe miały na celu przetestować przypadki, w których nazwa trasy wybrana przez turystę jest dla niego unikalna lub też nie (tzn. czy turysta ma już trasę o podanej nazwie, czy też nie). W tym celu wysyłane jest zapytanie POST, które pośrednio wywołuje metodę `get_if_tour_name_unique` z pakietu services (to właśnie poprawność działania tej metody jest testowana). Następnie sprawdzany jest status odpowiedzi oraz informacja zwrotna są takie, jakich oczekiwaliśmy.

W screenach testów oraz ich wyników pozwoliłam sobie zamazać ścieżki na moim dysku.

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

Testy jednostkowe dla dodawania punktu własnego (Justyna Małuszyńska)

```

5   # common case
6   # testing /own-points POST endpoint for new point with correct values
7   # this endpoint should return message and status code 200
8 def test_add_point_with_correct_values(client):
9     new_point_data = {
10       "name": "Smerek",
11       "latitude": 18.1373,
12       "longitude": 23.7853
13     }
14
15     response = client.post(
16       "/own-points",
17       data = json.dumps(new_point_data),
18       headers = {"Content-Type": "application/json"},
19     )
20
21     message = json.loads(response.data.decode('utf-8')).get("message")
22
23     assert response.status_code == 200
24     assert message == "Punkt został pomyślnie dodany"
25

```

Powyższy test jednostkowy miał na celu przetestować przypadek, w którym turysta chciałby dodać nowy, nieistniejący punkt własny do bazy, podając przy tym poprawne dane (unikalną nazwę oraz unikalne współrzędne geograficzne). W tym celu tworzony jest więc nowy punkt, następnie wysypane jest zapytanie POST, a odpowiedź zapytania porównuje się z oczekiwanymi wartościami, czyli statusem odpowiedzi 200 oraz informacją zwrotną. Endpoint /own-points dla akcji POST wywołuje metodę add_own_point z pakietu services i to właśnie poprawność działania tej metody jest testowana.

```

93  # edge case
94  # testing /own-points POST endpoint for new point with non unique coordinates
95  # this endpoint should return message with error and status code 400
96 def test_add_point_with_non_unique_coordinates(client):
97   new_point = {
98     "name": "Pieskowa Skała",
99     "latitude": 18.1373,
100    "longitude": 23.7853
101  }
102
103  response = client.post(
104    "/own-points",
105    data = json.dumps(new_point),
106    headers = {"Content-Type": "application/json"},
107  )
108
109  message = json.loads(response.data.decode('utf-8')).get("message")
110
111  assert response.status_code == 400
112  assert message == "Punkt z podanymi współrzędnymi geograficznymi już istnieje"
113

```

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

Powyższy test jednostkowy miał na celu przetestować przypadek, w którym turysta chciałby dodać nowy, nieistniejący punkt własny do bazy, podając przy tym niepoprawne dane – nieunikalne współrzędne geograficzne. W tym celu tworzony jest więc nowy punkt (z podanymi współrzędnymi, które istnieją już w bazie), następnie wysyłane jest zapytanie POST, a odpowiedź zapytania porównuje się z oczekiwanyymi wartościami, czyli statusem odpowiedzi 400 oraz informacją zwrotną o występującym błędzie.

```

49  # edge case
50  # testing /own-points POST endpoint for new point with incorrect latitude
51  # this endpoint should return message with error and status code 400
52  def test_add_point_with_incorrect_latitude(client):
53      new_point = {
54          "name": "Pieskowa Skała",
55          "latitude": "23.",
56          "longitude": 23.7853
57      }
58
59      response = client.post(
60          "/own-points",
61          data = json.dumps(new_point),
62          headers = {"Content-Type": "application/json"},
63      )
64
65      message = json.loads(response.data.decode('utf-8')).get("message")
66
67      assert response.status_code == 400
68      assert message == "Niepoprawne dane. Szerokość geograficzna musi być liczbą rzeczywistą"
69

```

Powyższy test jednostkowy miał na celu przetestować przypadek, w którym turysta chciałby dodać nowy, nieistniejący punkt własny do bazy, podając przy tym niepoprawne dane – szerokość geograficzna w niepoprawnym formacie („23.”). W tym celu tworzony jest więc nowy punkt z błędnią szerokością geograficzną, następnie wysyłane jest zapytanie POST, a odpowiedź zapytania porównuje się z oczekiwanyymi wartościami, czyli statusem odpowiedzi 400 oraz informacją zwrotną o występującym błędzie.

W pliku testów pobierania danych punktu własnego (backend\tests\ test_own_pointsAPI_add.py) znajduje się jeszcze kilka testów działających na podobnej zasadzie, testujących dodawanie punktu z niepoprawnymi formatami danych, z nieunikalną nazwą.

Screen potwierdzający działanie testów:

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

Testy jednostkowe dla edytowania punktu własnego (Justyna Małuszyńska)

```

5   # common case
6   # testing /own-points/<int:id> PUT endpoint for point - correct new data
7   # this endpoint should return message and status code 200
8 def test_update_point_correct_new_data(client):
9     update_point_id = 21
10    new_data_point = {
11        "name": "Kopa",
12        "latitude": 64.2345,
13        "longitude": 16.5765
14    }
15
16    response = client.put(
17        f"/own-points/{update_point_id}",
18        data=json.dumps(new_data_point),
19        headers={"Content-Type": "application/json"},
20    )
21
22    message = json.loads(response.data.decode('utf-8')).get("message")
23
24    assert response.status_code == 200
25    assert message == "Punkt został pomyślnie edytowany"
26

```

Powyższy test jednostkowy miał na celu przetestować przypadek, w którym turysta chciałby edytować punkt własny, podając przy tym poprawne dane (unikalna nazwa, unikalne i w odpowiednim formacie współrzędne geograficzne oraz punkt, którego dane są zmieniane, nie jest używany w odcinkach). W tym celu tworzony jest więc nowy punkt z nowymi danymi, następnie wysyłane jest zapytanie PUT, a odpowiedź zapytania porównuje się z oczekiwany wartościami, czyli statusem odpowiedzi 200 oraz informacją zwrotną. Warto dodać, że w bazie nie ma punktu o nazwie Kopa przypisanego do testowanego turysty, a punkt o id 21 nie jest używany w odcinkach. Endpoint /own-points dla akcji PUT korzysta z metody update_own_point z pakietu services i to właśnie poprawność działania tej metody jest testowana.

```

215 # edge case
216 # testing /own-points/<int:id> PUT endpoint for point where new coordinates are already taken by another point
217 # this endpoint should return message with error and status code 400
218 def test_update_point_with_existing_coordinates(client):
219     update_point_id = 4
220     new_point_data = {
221         "name": "Gubałówka",
222         "latitude": 94.4353,
223         "longitude": 21.2342
224     }
225
226     response = client.put(
227         f"/own-points/{update_point_id}",
228         data=json.dumps(new_point_data),
229         headers={"Content-Type": "application/json"},
230     )
231
232     message = json.loads(response.data.decode('utf-8')).get("message")
233
234     assert response.status_code == 400
235     assert message == "Punkt z podanymi współrzędnymi geograficznymi już istnieje"
236

```

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

Powyższy test jednostkowy miał na celu przetestować przypadek, w którym turysta chciałby edytować punkt własny, podając przy tym nieunikalne wymagane dane (nieunikalne współrzędne geograficzne). W tym celu tworzony jest więc nowy punkt z testowymi danymi, następnie wysyłane jest zapytanie PUT, a odpowiedź zapytania porównuje się z oczekiwanymi wartościami, czyli statusem odpowiedzi 400 oraz informacją zwrotną o błędzie. Warto dodać, że w bazie jest już punkt o id 4, który nie jest używany w odcinkach.

```

238 # edge case
239 # testing /own-points/<int:id> PUT endpoint for point that can't be updated because it is already being used
240 # this endpoint should return message with error and status code 400
241 def test_update_point_that_can_not_be_updated(client):
242     update_point_id = 8
243     new_point_data = {
244         "name": "Nazwa",
245         "latitude": 21.3333,
246         "longitude": 3.3333
247     }
248
249     response = client.put(
250         f"/labeled-points/{update_point_id}",
251         data=json.dumps(new_point_data),
252         headers={"Content-Type": "application/json"},
253     )
254
255     message = json.loads(response.data.decode('utf-8')).get("message")
256
257     assert response.status_code == 400
258     assert message == "Punkt jest już używany w odcinkach. Nie można go edytować"
259

```

Powyższy test jednostkowy miał na celu przetestować przypadek, w którym turysta chciałby edytować punkt własny, który jest już używany w odcinkach. W tym celu tworzony jest więc nowy punkt z testowymi danymi, następnie wysyłane jest zapytanie PUT, a odpowiedź zapytania porównuje się z oczekiwanyimi wartościami, czyli statusem odpowiedzi 400 oraz informacją zwrotną o błędzie. Warto dodać, że w bazie punkt o id 8 jest punktem przypisany do odcinków.

W pliku testów pobierania danych punktu własnego (backend\tests\ test_own_pointsAPI_update.py) znajduje się jeszcze kilka testów działających na podobnej zasadzie, testujących edycję punktu z niepoprawnymi formatami danych, punktu nieistniejącego, z nieunikalną nazwą.

Screen potwierdzający działanie testów:

```

    ✓ ⏺ test_own_pointsAPI_update.py
      ✓ test_update_point_correct_new_data
      ✓ test_update_point_for_the_same_name
      ✓ test_update_point_for_the_same_coordinates
      ✓ test_update_used_point_for_the_same_name
      ✓ test_update_point_with_existing_name_and_incorrect_latitude
      ✓ test_update_point_with_existing_name_and_incorrect_longitude
      ✓ test_update_point_with_invalid_longitude
      ✓ test_update_point_with_invalid_latitude
      ✓ test_update_point_with_existing_name
      ✓ test_update_point_with_existing_coordinates
      ✓ test_update_point_that_can_not_be_updated
      ✓ test_update_point_that_not_exists

```

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

Testy jednostkowe dla pobierania danych punktu własnego (Justyna Maluszyńska)

```

5   # common case
6   # testing /own-points/<int:id> GET endpoint for point with specified id
7   # this endpoint should return requested own point data and status code 200
8 def test_get_specified_id(client):
9     own_point_id = 9
10
11    response = client.get(f"/own-points/{own_point_id}")
12    own_point = json.loads(response.data.decode('utf-8'))
13    own_point_returned_id = own_point['id']
14
15    assert response.status_code == 200
16    assert own_point_returned_id == own_point_id
17

```

Powyższy test jednostkowy miał na celu przetestować przypadek, w którym aplikacja potrzebuje danych konkretnego punktu, podając id punktu. W tym celu wysyłane jest zapytanie GET z odpowiednim id punktu (istniejącym w bazie i należącym do zalogowanego turysty), a odpowiedź zapytania porównuje się z oczekiwanyymi wartościami, czyli statusem odpowiedzi 200 oraz id zwracanego punktu bazy. Endpoint /own-points/<id:int> dla akcji GET korzysta z metody get_own_point z pakietu services i to właśnie poprawność działania tej metody jest testowana.

```

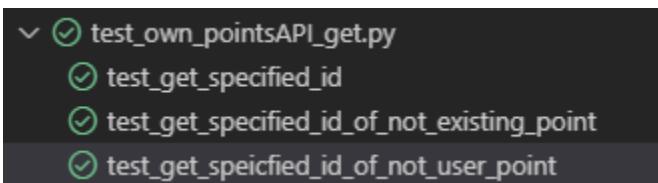
32  # edge case
33  # testing /own-points/<int:id> GET endpoint for point with specified id
34  # requested point do not belongs to logged in user
35  # this endpoint should return message with error and status code 400
36 def test_get_specified_id_of_not_user_point(client):
37     own_point_id = 19
38
39     response = client.get(f"/own-points/{own_point_id}")
40     message = json.loads(response.data.decode('utf-8')).get("message")
41
42     assert response.status_code == 400
43     assert message == 'Punkt nie należy do tego użytkownika'

```

Powyższy test jednostkowy miał na celu przetestować przypadek, w którym aplikacja potrzebuje danych konkretnego punktu, podając id punktu nienależącego do zalogowanego turysty. W tym celu wysyłane jest zapytanie GET z odpowiednim id punktu (istniejącym w bazie i nienależącym do zalogowanego turysty), a odpowiedź zapytania porównuje się z oczekiwanyymi wartościami, czyli statusem odpowiedzi 400 oraz informacja zwrotna o błędzie.

W pliku testów pobierania danych punktu własnego (backend\tests\ test_own_pointsAPI_get.py) znajduje się jeszcze jeden test, testujący pobieranie danych punktu nieistniejącego w bazie.

Screen potwierdzający działanie testów:



Testy jednostkowe dla sprawdzania czy przewodnik istnieje w systemie (Justyna Maluszyńska)

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

```

5   # common case
6   # testing /evidence-confirmation/guide/<int:id_guide> GET endpoint for checking if requested guide exist - guide with specified id exists
7   # this endpoint should return message and status code 200
8 def test_check_guide_exists(client):
9     id_number_guide = 123456
10
11    response = client.get(f"/evidence-confirmation/guide/{id_number_guide}")
12
13    message = json.loads(response.data.decode('utf-8')).get("message")
14
15    assert response.status_code == 200
16    assert message == "OK"

```

Powyższy test jednostkowy miał na celu przetestować przypadek, w którym aplikacja potrzebuje zweryfikować czy przewodnik o podanym numerze legitymacji przez użytkownika istnieje. W tym celu wysyłane jest zapytanie GET z podanym przez użytkownika numerze legitymacji (istniejącym w bazie), a odpowiedź zapytania porównuje się z oczekiwanyymi wartościami, czyli statusem odpowiedzi 200 oraz wiadomością zwrotną. Endpoint /evidence-confirmation/guide/<int:id_guide> dla akcji GET korzysta z metody check_if_guide_exists z pakietu services i to właśnie poprawność działania tej metody jest testowana.

```

19 # edge case
20 # testing /evidence-confirmation/guide/<int:id_guide> GET endpoint for checking if requested guide exist - guide with specified id do not exists
21 # this endpoint should return message and status code 200
22 def test_check_guide_that_not_exists(client):
23     id_number_guide = 654321
24
25     response = client.get(f"/evidence-confirmation/guide/{id_number_guide}")
26
27     message = json.loads(response.data.decode('utf-8')).get("message")
28
29     assert response.status_code == 404
30     assert message == "Przewodnik o podanym numerze legitymacji nie istnieje"

```

Powyższy test jednostkowy przebiega identycznie jak wymieniony wyżej, natomiast podawany jest numer legitymacji przewodnika który nie istnieje w bazie. Wysyłane jest zapytanie GET z podanym przez użytkownika numerze legitymacji (nieistniejącym w bazie), a odpowiedź zapytania porównuje się z oczekiwanyymi wartościami, czyli statusem odpowiedzi 404 oraz informacją zwrotną o błędzie.

Screen potwierdzający działanie testów:

```

v )test_evidence_confirmationAPI_check_guide.py
  ✓ test_check_guide_exists
  ✓ test_check_guide_that_not_exists

```

Testy jednostkowe dla dodawania dowodów do weryfikacji (Justyna Maluszyńska)

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

```

5   # common case
6   # testing /evidence-confirmation/evidence POST endpoint for adding new evidences to confirm selected tour segments
7   # in this case, user post few segments with attachment and few segemnts with id number of verifying guide
8   # this endpoint should return array of reported tour segments (with evidence data) and status code 200
9 def test_add_evidence_attachment_and_id_guide(client):
10     final_segments = [
11         {"id": 1, "attachment": "inny_dowod.png", "startDate": "2020-01-01", "endDate": "2020-01-01"},
12         {"id": 2, "attachment": "dowod.png", "startDate": "2020-01-01", "endDate": "2020-01-02"},
13         {"id": 3, "attachment": "dowod.png", "startDate": "2020-01-03", "endDate": "2020-01-03"},
14         {"id": 4, "verifying": 123456, "startDate": "2020-01-04", "endDate": "2020-01-04"},
15         {"id": 5, "verifying": 123456, "startDate": "2020-01-05", "endDate": "2020-01-05"}
16     ]
17
18     tour_segments_data = {
19         "attachments": [
20             [
21                 {"value": "dowod.png", "mountainGroup": 3, "tourSegments": [
22                     [
23                         {"id": 2, "startDate": "2020-01-01", "endDate": "2020-01-02"}, {"id": 3, "startDate": "2020-01-03", "endDate": "2020-01-03"}]
24                 ],
25                 {"value": "inny_dowod.png", "mountainGroup": 3, "tourSegments": [
26                     [
27                         {"id": 1, "startDate": "2020-01-01", "endDate": "2020-01-01"}]
28                 ],
29             ],
30         ],
31         "verifying": [
32             {"idVerifying": 123456, "tourSegments": [
33                 [
34                     {"id": 4, "startDate": "2020-01-04", "endDate": "2020-01-04"}, {"id": 5, "startDate": "2020-01-05", "endDate": "2020-01-05"}]
35             ],
36         ]
37     }
38 }
39
40     response = client.post(
41         "/evidence-confirmation/evidence",
42         data=json.dumps(tour_segments_data),
43         headers={"Content-Type": "application/json"},
44     )
45
46     assert response.status_code == 200
47
48     segments = json.loads(response.data.decode('utf-8'))
49     for segment in segments:
50         tour_segment = next(item for item in final_segments if item["id"] == segment["id"])
51         assert tour_segment["startDate"] == segment["startDate"]
52         assert tour_segment["endDate"] == segment["endDate"]
53         if tour_segment.get("attachment") != None:
54             assert tour_segment.get("attachment") == segment["evidence"].get("photo_attachment")
55         else:
56             assert tour_segment.get("verifying") == segment["evidence"]["verifying"]["guide"]["id_number"]
57

```

Ze względu na to, jak dla przypadku użycia został zaimplementowany interfejs użytkownika, niemożliwym jest podanie dat w nieprawidłowym formacie, czy też podanie daty początku późniejszej od daty końca odcinka trasy. Z tego powodu testowanie funkcji ogranicza się do zweryfikowania, czy daty zostały poprawnie przypisane do każdego z podanych odcinków tras oraz zweryfikowanie czy każdy z odcinków został przypisany do odpowiedniego dowodu.

Na początku tworzymy listę zawierającą oczekiwane rezultaty zwrócone przez naszą testowaną metodę. Następnie symulujemy przekazane dane od użytkownika (rodzaj dowodu, odcinki trasy, daty przebycia odcinków). Wysypane jest zapytanie POST z zasymulowanymi danymi, a odpowiedź zapytania porównuje się z oczekiwanyymi wartościami, czyli statusem odpowiedzi 200 oraz danymi zwrotnymi o odcinkach i dowodach.

Endpoint /evidence-confirmation/evidence dla akcji POST korzysta z metody add_evidences z pakietu services i to właśnie poprawność działania tej metody jest testowana.

“Turysta Z Odznaką”<Project Name>	
Etap I	Data: <dd/mmm/yy>

W pliku testów zgłaszania do weryfikacji trasy (backend\tests\test_evidence_confirmationAPI_add_evidence.py) znajduje się drugi test, działający na podobnej zasadzie.

Screen potwierdzający działanie testów:

- ✓ `test_evidence_confirmationAPI_add_evidence.py`
- ✓ `test_add_evidence_attachment_and_id_guide`
- ✓ `test_add_evidence_only_attachment`

4. Ocena jakości kodu

Ocena jakości kodu nie została wykonana