a) „Platforma do organizacji wydarzeń” (EN: "Platform for organizing events") to projekt informatyczny, który ma na celu stworzenie kompleksowej platformy umożliwiającej organizację różnego rodzaju wydarzeń. Platforma ta ma być wszechstronnym narzędziem, które ułatwi zarządzanie, planowanie i promocję różnego rodzaju imprez, począwszy od małych spotkań lokalnych, po duże konferencje czy festiwale. Oto ogólny opis projektu:

b) Cel projektu:

Stworzenie kompleksowej platformy internetowej umożliwiającej łatwą organizację, zarządzanie i różnorodnych wydarzeń.

c) przedział czasowy jaki będzie konieczny do realizacji projektu

W przypadku idealnym: początek czerwca, ale zakładam, że koniec czerwca.

d) Spis członków:

Mikhail Yakushevich – Team Leader. Zadania: https://mikhail-yakushevich.atlassian.net/issues/?filter=10003

Hubert Moś – Programista/Tester. Zadania: https://mikhail-yakushevich.atlassian.net/issues/?filter=10005

Olaf Myszak – Programista/Tester. Zadania: https://mikhail-yakushevich.atlassian.net/issues/?filter=10004

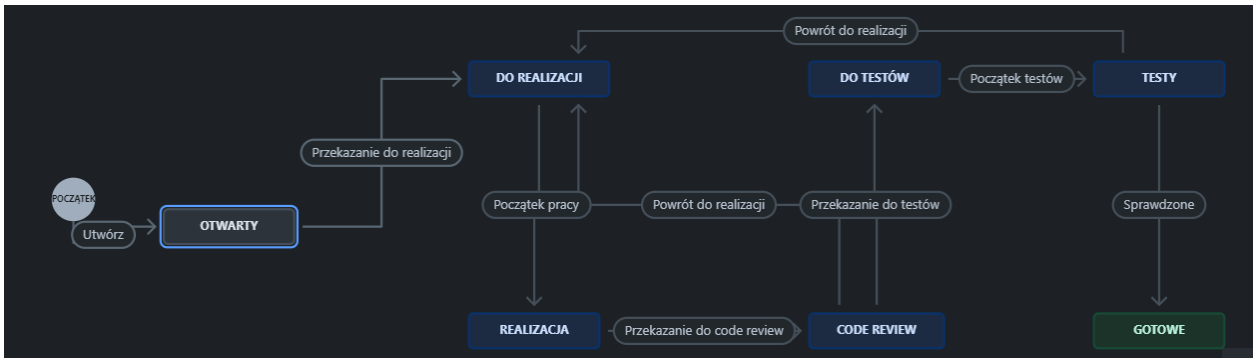
spis członków projektu wraz z ich zakresem obowiązków oraz **listą zadań jakie** wykonali w cyklu pracy nad projektem (najlepiej link do JIRA)

e) spis kroków milowych (milestone: 2-3 mce), epików (2-4 tyg) oraz tasków /

historyjek / pod-zadań wykonywanych w ramach historyjek (1 sprint)

https://mikhail-yakushevich.atlassian.net/jira/software/projects/PFOE/boards/2/timeline

Przepływ pracy(Jira):



f) opis i motywacja dla wyboru technologii wykorzystywanych w projekcie

W projekcie "Platforma do Organizacji Wydarzeń" zdecydowano się na wykorzystanie technologii C# w połączeniu z ASP.NET Core Web API po stronie serwera, oraz frameworku Angular po stronie klienta. Poniżej przedstawiamy opis oraz motywację dla tego wyboru technologicznego:

1. C# / ASP.NET Core Web API:

- Język Programowania C#:

- C# jest językiem programowania, który charakteryzuje się prostotą, efektywnością i bezpieczeństwem. Jest szczególnie dobrze zintegrowany z platformą .NET, co ułatwia rozwijanie skalowalnych aplikacji biznesowych.

- Jako język typowo obiektowy, C# ułatwia organizację i zarządzanie kodem, co jest istotne w projektach o większej skali.

- ASP.NET Core Web API:

- ASP.NET Core Web API jest frameworkiem stworzonym do tworzenia interfejsów programistycznych dla aplikacji internetowych. Jest lekki, wydajny i obsługuje model architektoniczny RESTful, co jest idealne dla projektów opartych na mikroserwisach.

- Zapewnia elastyczność w obszarze integracji z różnymi technologiami i formatami danych, co jest kluczowe w przypadku systemu obsługującego różnorodne funkcje związane z organizacją wydarzeń.

- Dodatkowo będzie zaimplementowana architektura Kontroler-Serwis-Repozytorium (Controller-Serwis-Repository):

1. Kontroler (Controller):

Kontroler jest warstwą interfejsu użytkownika i jest odpowiedzialny za obsługę żądań HTTP, interpretowanie danych wejściowych i decydowanie, jakie akcje mają zostać podjęte.

W przypadku architektury API, kontroler zajmuje się obsługą żądań przychodzących od klientów i decyduje, jakie akcje powinny zostać podjęte.

Kontroler nie zawiera logiki biznesowej. Jego głównym zadaniem jest koordynacja przepływu sterowania i przekazywanie żądań do warstwy usług.

2. Serwis (Service):

Warstwa serwisu zawiera logikę biznesową aplikacji. To tutaj są umieszczone operacje, które nie powinny być bezpośrednio w kontrolerze, aby zachować zasady jednej odpowiedzialności (Single Responsibility Principle).

Serwisy są odpowiedzialne za interakcję z repozytorium (lub innymi źródłami danych), przetwarzanie danych, wykonanie operacji biznesowych i przygotowanie danych do zwrócenia lub zapisania w repozytorium.

Serwisy są często używane, gdy ta sama logika biznesowa jest współdzielona pomiędzy różnymi kontrolerami.

3. Repozytorium (Repository):

Repozytorium jest warstwą dostępu do danych. Odpowiada za operacje związane z pobieraniem, zapisywaniem, aktualizacją i usuwaniem danych w bazie danych lub innym źródle danych.

Separacja repozytorium od warstwy serwisu umożliwia elastyczne zarządzanie źródłami danych i ułatwia testowanie jednostkowe.

Repozytoria ukrywają szczegóły implementacyjne dostępu do danych przed serwisami i kontrolerami.

2. Angular:

- Dynamiczne Interfejsy Użytkownika:

- Angular, będący frameworkiem front-endowym, pozwala na łatwe tworzenie dynamicznych, responsywnych interfejsów użytkownika. To kluczowe w przypadku platformy do organizacji wydarzeń, gdzie estetyka i intuicyjność są istotne dla użytkowników.

- Dzięki architekturze opartej na komponentach, Angular ułatwia rozwój modułów i komponentów, co przyspiesza proces tworzenia interfejsu.

- Język TypeScript:

- Angular współpracuje z TypeScriptem, co pozwala na bardziej zorganizowany i bezpieczny rozwój aplikacji. Typowanie statyczne TypeScriptu pomaga w identyfikacji błędów już na etapie pisania kodu.

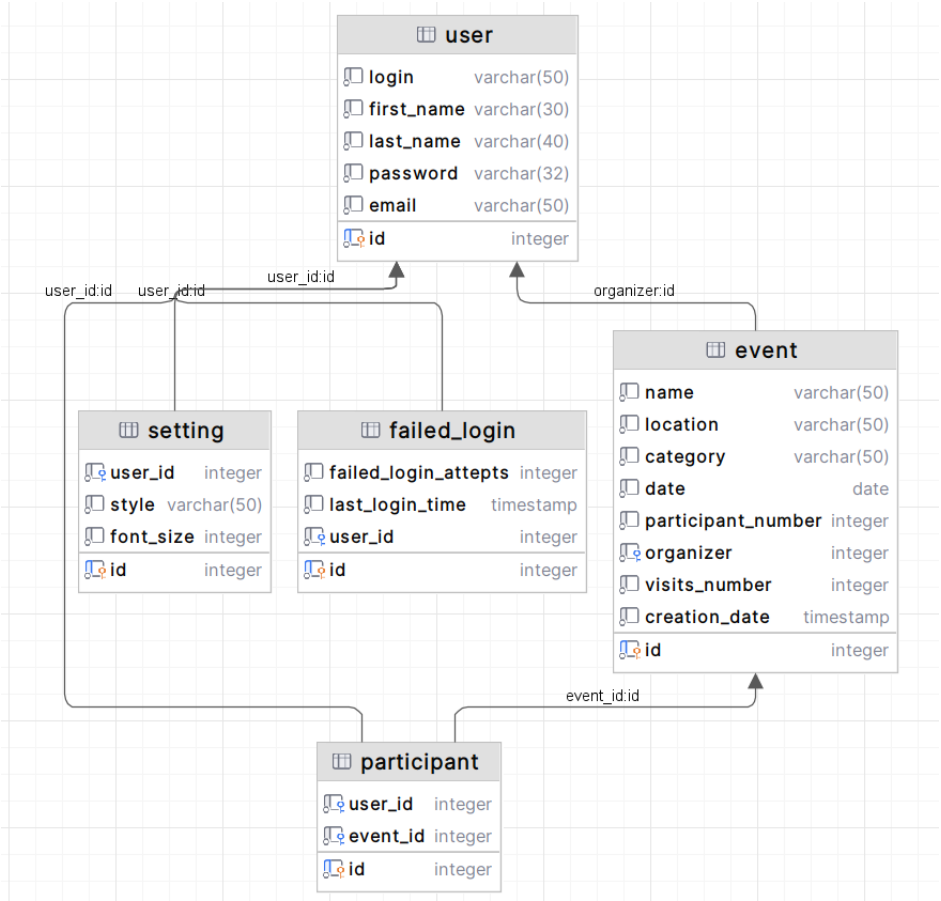
- Modularność i Reużywalność:

- Angular wspiera modułowość, co pozwala na łatwe zarządzanie kodem i zwiększa jego ponowne wykorzystanie. To istotne w projektach, gdzie struktura aplikacji musi być elastyczna i skalowalna.

Wybór C# / ASP.NET Core Web API i Angular jest uzasadniony ich zdolnością do tworzenia wydajnych, skalowalnych i modularnych rozwiązań, co jest kluczowe dla udanej implementacji platformy do organizacji wydarzeń.

g) dokumentacja techniczna

• schemat bazy danych, PostgreSQL



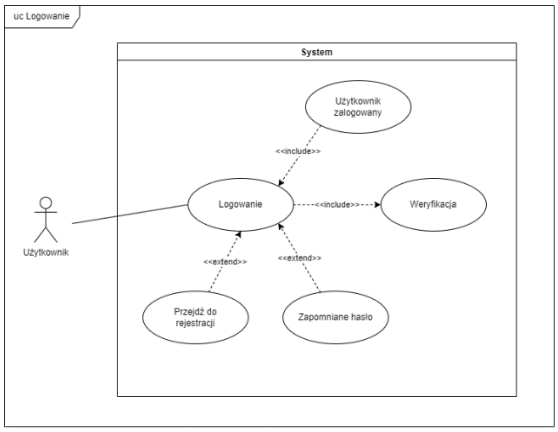
Źródło:

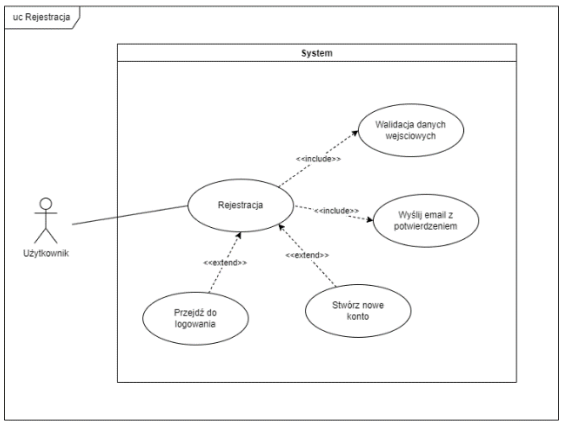
- XML: docs/xml/database_schema.xml
- PNG: docs/png/database_schema.png

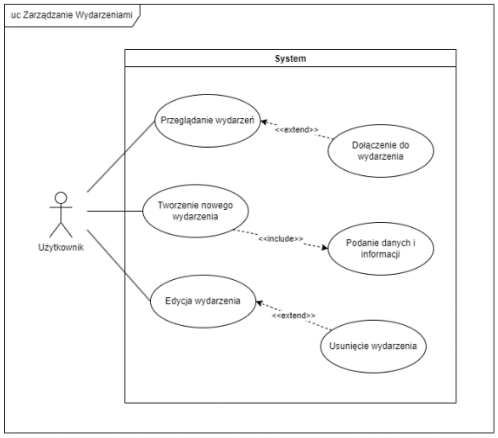
Zapytania:

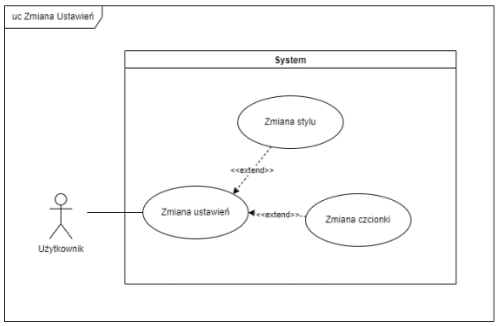
1. **Dodawanie nowego użytkownika:**
   INSERT INTO "user" (login, first_name, last_name, password, email) VALUES ('nowy_uzytkownik', 'Imie', 'Nazwisko', '1a7fcdd5a9fd433523268883cfded9d0', 'email@example.com');
2. **Pobieranie informacji o użytkowniku na podstawie loginu:**
   SELECT * FROM "user" WHERE login = 'szukany_login';
3. **Dodawanie nowego wydarzenia:**
   INSERT INTO event (name, location, category, date, participant_number, organizer, visits_number, creation_date) VALUES ('Nowe Wydarzenie', 'Lokacja', 'Kategoria', '2024-01-30', 100, 1, 0, NOW());
4. **Pobieranie listy wydarzeń zorganizowanych przez danego użytkownika:**
   SELECT * FROM event WHERE organizer = 1;
5. **Dodawanie nowego uczestnika do wydarzenia:**
   INSERT INTO participant (user_id, event_id) VALUES (2, 1);

• diagramy przypadków użycia





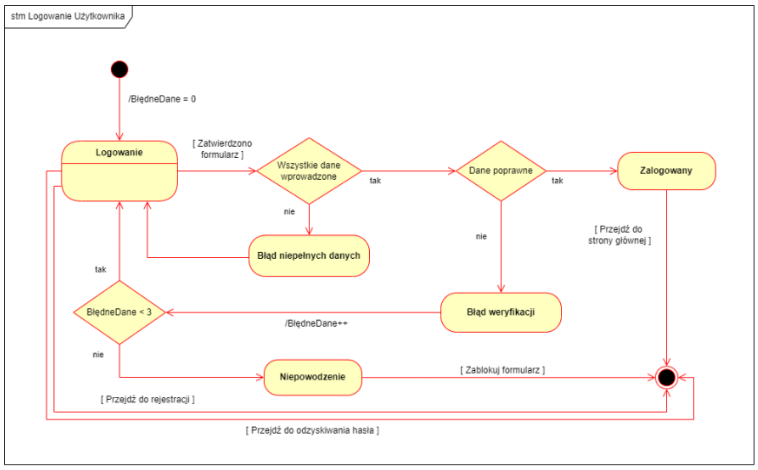


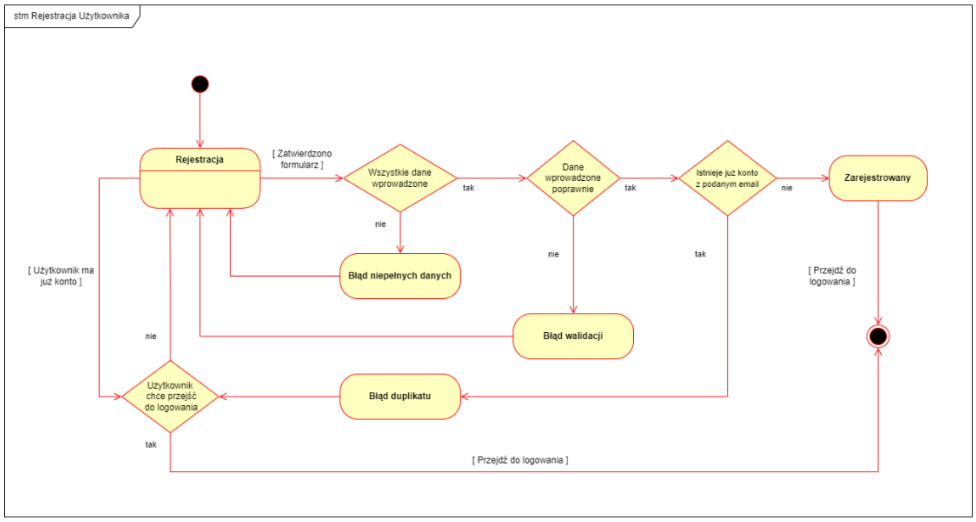


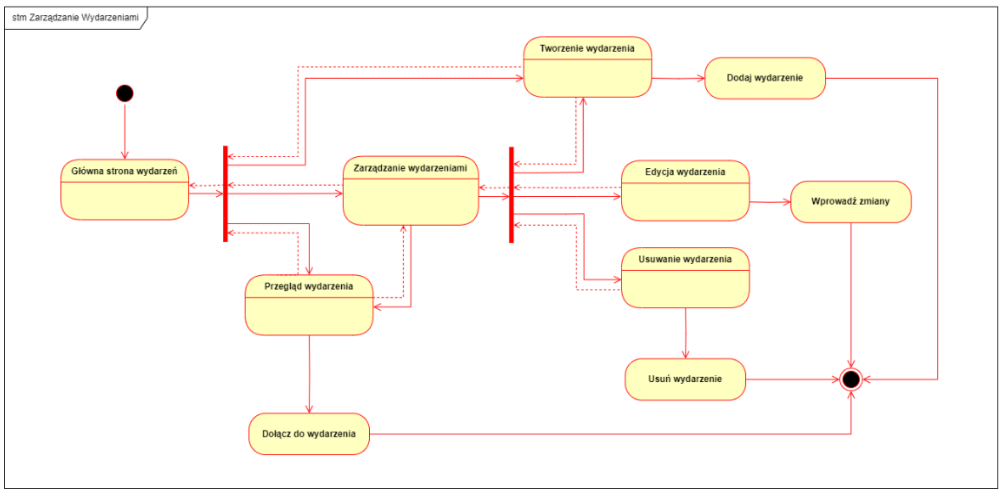
Źródło:

- XML: docs/xml/diagramy_przypadkow.xml
- PNG: docs/png/diagramy_przypadkow.png

• diagramy stanów



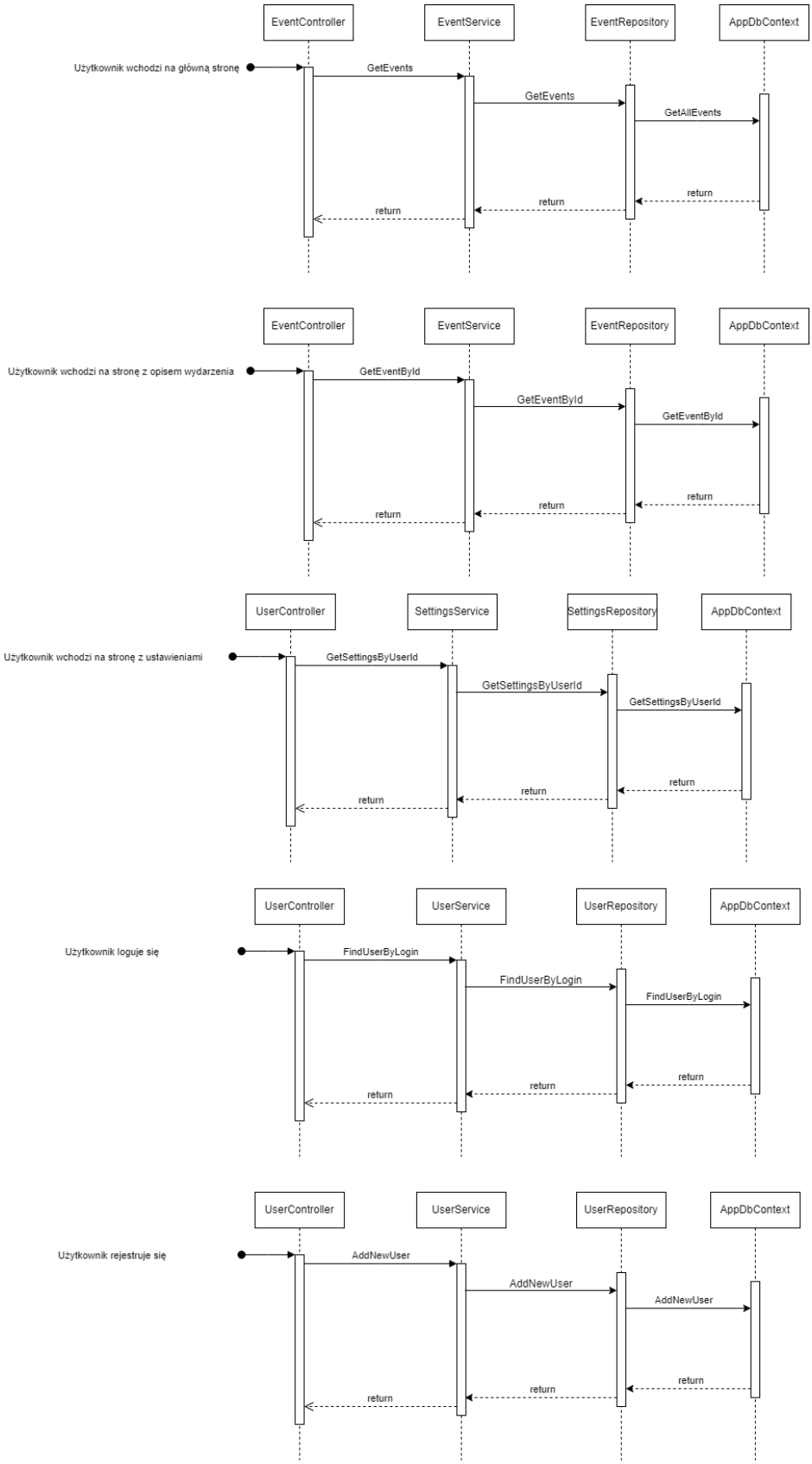




Źródło:

- XML: docs/xml/diagramy_stanow.xml
- PNG: docs/png/diagramy_stanow.png

• diagramy sekwencji dla głównych funkcjonalności

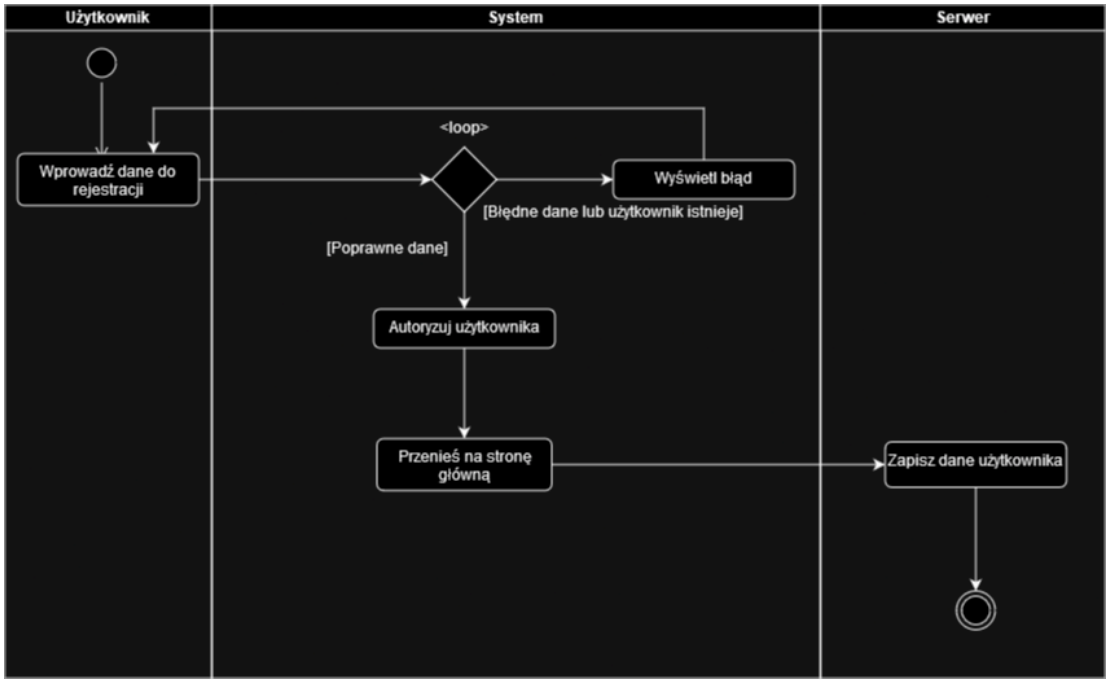


Źródło:

- XML: docs/xml/sequence_diagram.xml
- PNG: docs/png/sequence_diagram.png

• diagramy czynności

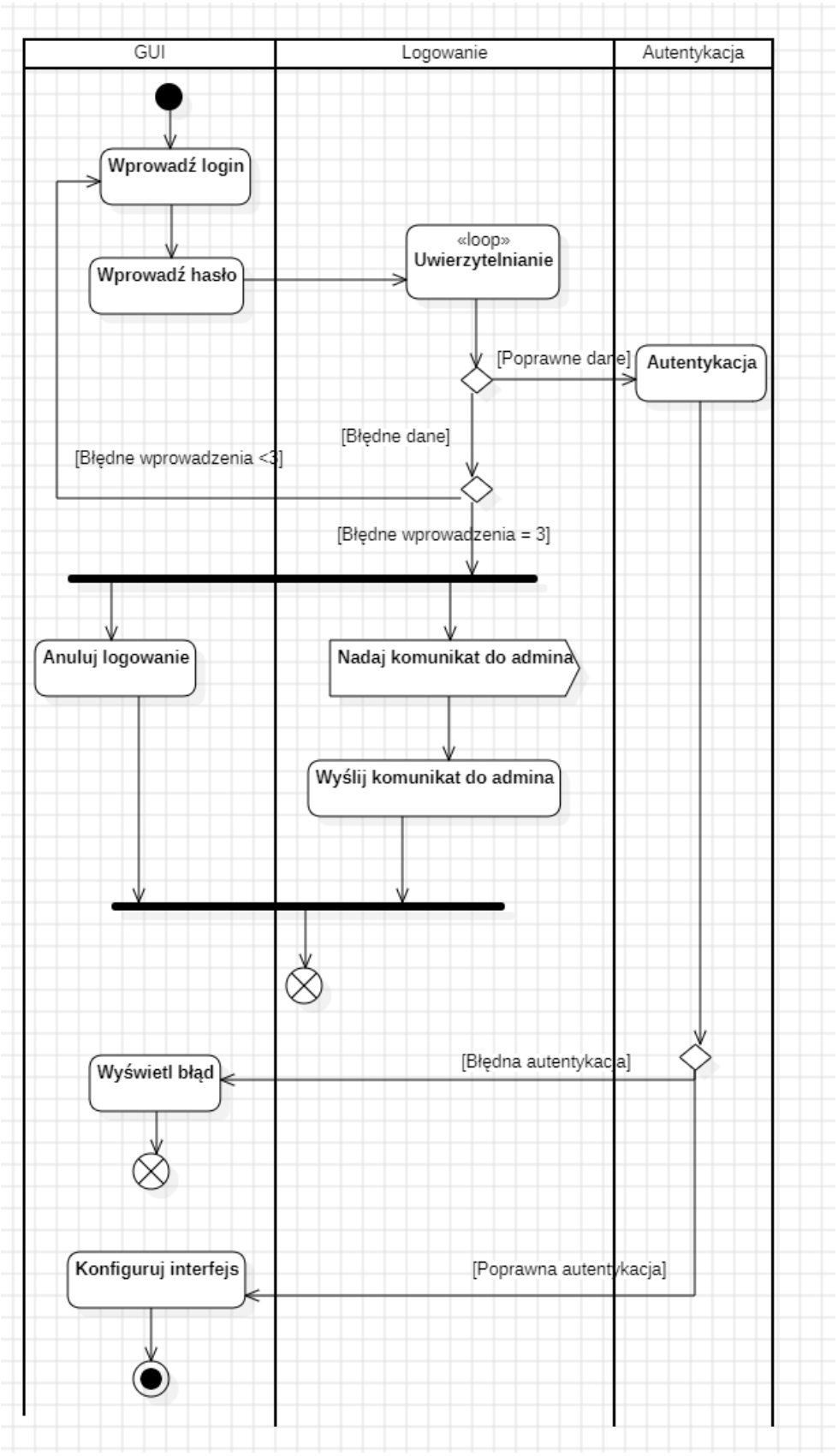
Rejestracja:



Źródło:

- XML: docs/xml/RegistrationActivityDiagram.xml
- PNG: docs/png/RegistrationActivityDiagram.png

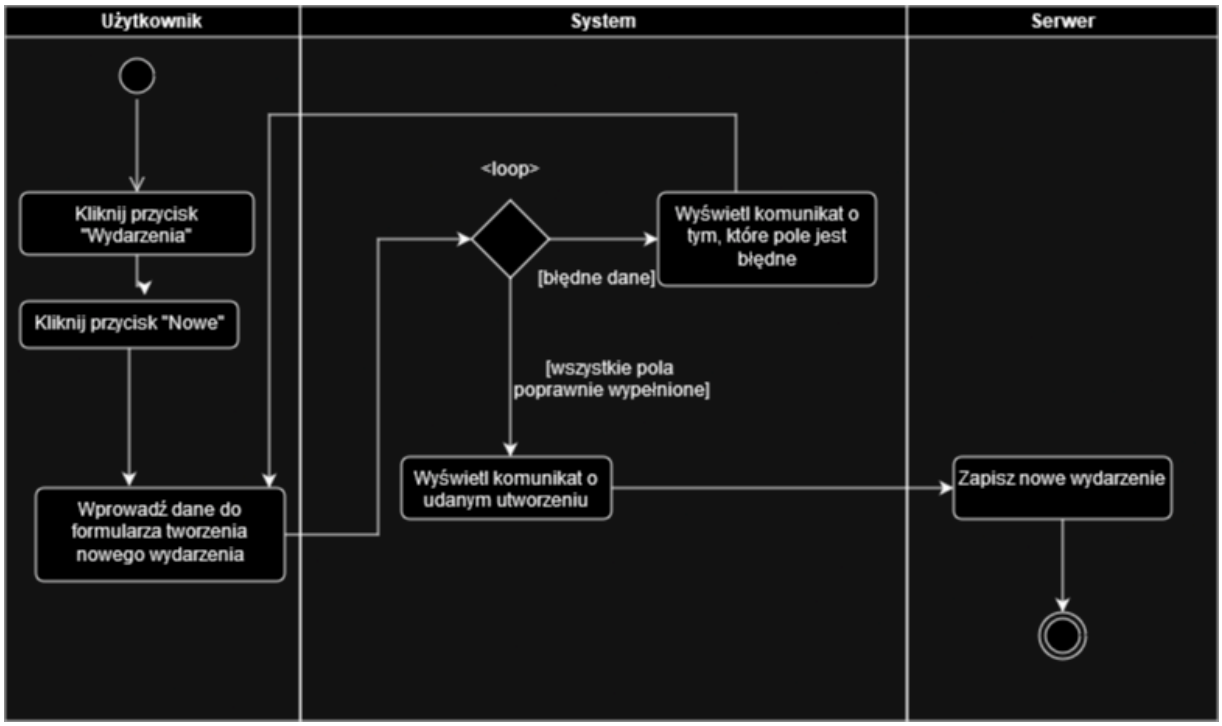
Logowanie:



Źródło:

- XML: docs/xml/LoginActivityDiagram.xml
- PNG: docs/png/LoginActivityDiagram.png

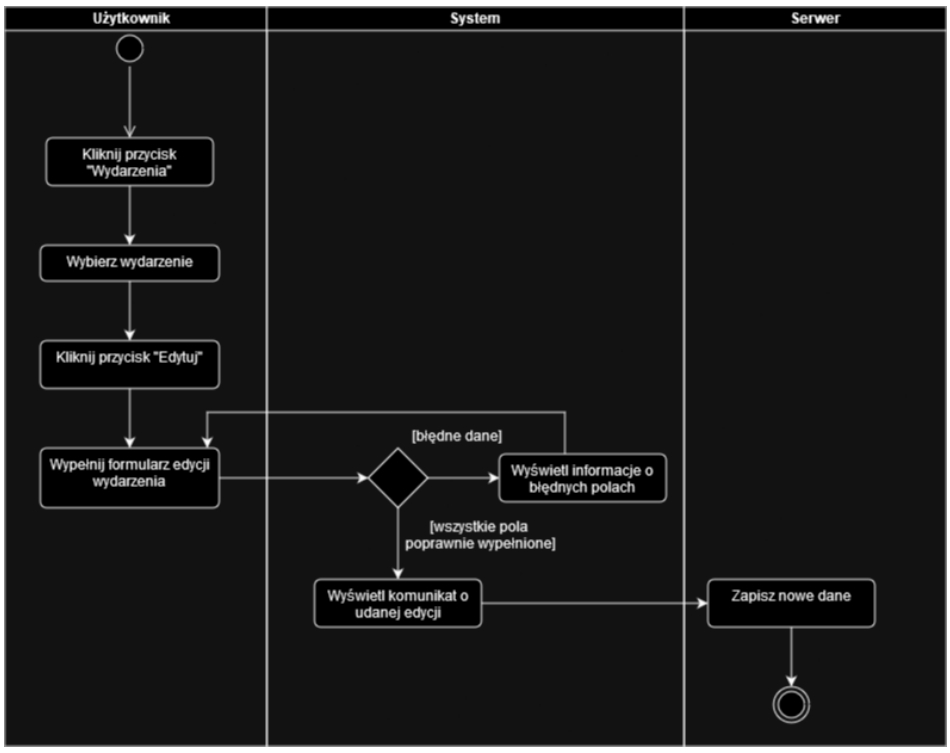
Tworzenie nowego wydarzenia:



Źródło:

- XML: docs/xml/NewEventActivityDiagram.xml
- PNG: docs/png/NewEventActivityDiagram.png

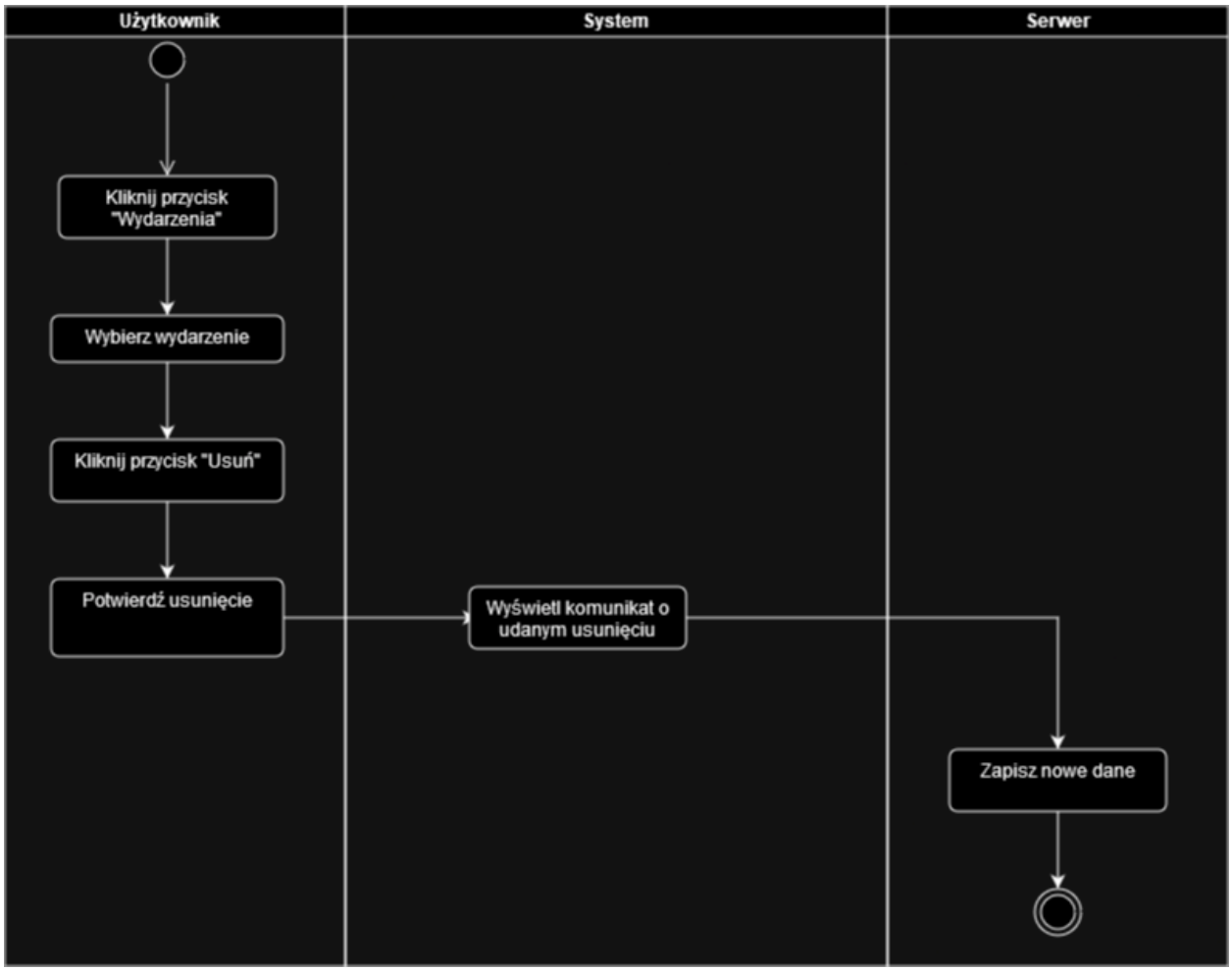
Edycja istniejącego wydarzenia:



Źródło:

- XML: docs/png/EditEventActivityDiagram.png
- PNG: docs/xml/EditEventActivityDiagram.xml

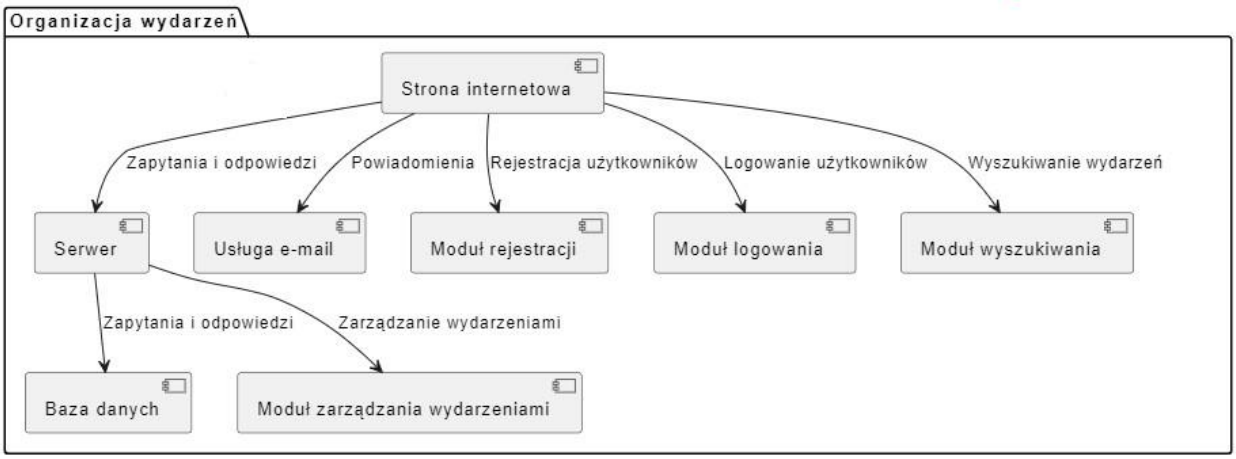
Usunięcie wydarzenia:



Źródło:

- XML: docs/xml/DeleteEventActivityDiagram.xml
- PNG: docs/png/DeleteEventActivityDiagram.png

Diagram komponentów:



Źródło:

- XML: docs/xml/components_diagram.xml
- PNG: docs/png/components_diagram.png

• scenariusze testowe (Given / When / Then)

1. Test rejestracji (poprawne dane)

Given: Użytkownik nie ma konta
When: Użytkownik podaje poprawne dane do rejestracji i klika przycisk “Zarejestruj się”
Then: System autoryzuje użytkownika i przenosi do strony głównej

2. Test rejestracji (nie zostało podane imię)

Given: Użytkownik nie ma konta
When: Użytkownik nie podaje imienia, ale podaje pozostałe dane
Then: System pokazuje błąd, że musi być podane imię

3. Test rejestracji (nie został podany e-mail)

Given: Użytkownik nie ma konta
When: Użytkownik nie podaje e-mail, ale podaje pozostałe dane
Then: System pokazuje błąd, że musi być podany e-mail

4. Test rejestracji (nie zostało podane nazwisko)

Given: Użytkownik nie ma konta
When: Użytkownik nie podaje nazwiska, ale podaje pozostałe dane
Then: System pokazuje błąd, że musii być podane nazwisko

5. Test rejestracji (nie zostało podane hasło)

Given: Użytkownik nie ma konta
When: Użytkownik nie podaje hasło, ale podaje pozostałe dane
Then: System pokazuje błąd, że musi być podane hasło

6. Test rejestracji (hasło zostało błędnie powtórzone)

Given: Użytkownik nie ma konta
When: Użytkownik nie podaję poprawnego powtórzenia hasła
Then: System pokazuje błąd, że hasła muszą być jednakowe

7. Test rejestracji na e-mail, który jest już użyty innym użytkownikiem

Given: Użytkownik nie ma konta
When: Użytkownik podaje dane razem z mailem, który jest przypisany do innego konta
Then: System pokazuje błąd, że użytkownik o podanym mailu już istnieje

8. Test logowania (błędny login)

Given: Użytkownik na stronie logowania
When: Użytkownik podaje błędny login
Then: System pokazuje pokazuje błąd, że zostały podane nieprawidłowe dane

9. Test logowania (błędne hasło)

Given: Użytkownik na stronie logowania
When: Użytkownik podaje błędne hasło
Then: System pokazuje pokazuje błąd, że zostały podane nieprawidłowe dane

10. Test logowania(poprawne dane)

Given: Użytkownik ma konto
When: Użytkownik podaje poprawny login oraz hasło
Then: System autoryzuje użytkownika i przenosi do strony głównej

11. Test przejścia na stronę wydarzeń

Given: Użytkownik zalogowany
When: Użytkownik klika w menu na zakładkę Wydarzenia
Then: Użytkownik zostaje przekierowany na stronę Wydarzeń

12. Test przejścia na stronę z opisem wydarzenia:

Given: Użytkownik zalogowany/niezalogowany
When: Klika w link z nazwą wydarzenia
Then: Użytkownik zostaje przekierowany na stronę z opisem wydarzenia

13. Test zapisu na wydarzenie:

Given: Użytkownik niezalogowany
When: Użytkownik próbuje się zapisać na wydarzenie
Then: Przycisk jest zablokowany

14. Test zapisu na wydarzenie:

Given: Użytkownik zalogowany
When: Użytkownik próbuje się zapisać na wydarzenie, gdzie nie ma już miejsc
Then: Przycisk jest zablokowany

15. Test zapisu na wydarzenie:

Given: Użytkownik zalogowany
When: Użytkownik próbuje się zapisać na wydarzenie, gdzie są miejsca
Then: Użytkownik dostaje komunikat, że zapis na wydarzenie został potwierdzony

16. Test tworzenia nowego wydarzenia

Given: Użytkownik na stronie Nowe Wydarzenie
When: Użytkownik uzupełnił formularz i klika przycisk “UTWÓRZ”
Then: Zostało utworzone nowe wydarzenie

17. Test tworzenia nowego wydarzenia(nie została podana nazwa)

Given: Użytkownik na stronie Nowe Wydarzenie
When: Użytkownik uzupełnił wszystkie pola, oprócz pola “Nazwa”
Then: Użytkownik dostaje komunikat, że musi być podana nazwa

18. Test tworzenia nowego wydarzenia(nie została podana kategoria)

Given: Użytkownik na stronie Nowe Wydarzenie
When: Użytkownik uzupełnił wszystkie pola, oprócz pola “Kategoria”
Then: Użytkownik dostaje komunikat, że musi być podana kategoria

19. Test tworzenia nowego wydarzenia(nie została podana data wydarzenia)

Given: Użytkownik na stronie Nowe Wydarzenie
When: Użytkownik uzupełnił wszystkie pola, oprócz pola “Data wydarzenia”
Then: Użytkownik dostaje komunikat, że musi być podana data wydarzenia

20. Test tworzenia nowego wydarzenia(nie została podana ilość miejsc)

Given: Użytkownik na stronie Nowe Wydarzenie
When: Użytkownik uzupełnił wszystkie pola, oprócz pola “Ilość miejsc”
Then: Użytkownik dostaje komunikat, że musi być podana ilość miejsc

21. Test usuwania wydarzenia

Given: Użytkownik próbuje usunąć swoje wydarzenie
When: Użytkownik wybiera wydarzenie, klika przycisk ‘Usuń’, pojawia się komunikat “Czy na pewno chcesz usunąć to wydarzenie?”(z możliwością wyboru “Tak”, czy “Nie”) i użytkownik klika przycisk “Tak”
Then: Wybrane wydarzenie zostało usunięte

22. Test edycji wydarzenia

Given: Użytkownik na stronie Edytuj Wydarzenie
When: Użytkownik wypełnił formularz
Then: Wydarzenie zostało edytowane

23. Test edycji wydarzenia (błędne dane)

Given: Użytkownik zalogowany na stronie Edytuj Wydarzenie
When: Użytkownik wypełnia formularz, podając błędne dane
Then: Zostają podświetlone na czerwono pola, w których użytkownik wprowadził błędne dane i przycisk zapisu zmian zostaje zablokowany

24. Test wylogowania się z konta

Given: Użytkownik zalogowany
When: Użytkownik klika przycisk ‘wyloguj’
Then: Użytkownik wylogowany i powrót do strony logowania

25. Test edycji ustawień konta

Given: Użytkownik zalogowany
When: Klika na swój obrazek, wybiera przycisk ‘Ustawienia konta’, przechodzi do strony do edycji swoich ustawień, zmienia ustawienia, które chcę i klika przycisk ‘Zapisz’
Then: Ustawienia zostają zaktualizowane i pojawi się komunikat o udanym zapisie

26. Test pola ‘Wyszukaj’ (dane istnieją)

Given: Użytkownik zalogowany/niezalogowany
When: Użytkownik podaje coś w polu ‘Wyszukaj’
Then: Pokazują się wyniki wyszukiwania wśród wydarzeń

27. Test pola ‘Wyszukaj’ (dane nie istnieją)

Given: Użytkownik zalogowany/niezalogowany
When: Użytkownik podaje coś w polu ‘Wyszukaj’
Then: Pokazują się informacja o nieznalezieniu danych wprowadzonych przez użytkownika

28. Test przycisku “Nie możesz się zalogować?”

Given: Użytkownik na stronie logowania
When: Użytkownik klika przycisk “Nie możesz się zalogować?
Then: Zostaje przeniesiony na stronę odzyskiwania konta

29. Test przejścia do okna Logowania

Given: Użytkownik na oknie Rejestracji
When: Użytkownik klika przycisk “Masz już konto? ‘Zaloguj się’”
Then: Wyświetl okno Logowania

30. Test przejścia do strony z opisem wydarzenia

Given: Użytkownik niezalogowany
When: Klika link z nazwą wydarzenia
Then: Wyświetla się strona z opisem wydarzenia

31. Test przejścia do okna Rejestracji

Given: Użytkownik na oknie Logowania
When: Użytkownik klika przycisk “Nie masz konta? ‘Zarejestruj się’”
Then: Wyświetl okno Rejestracji

32. Test przejścia do strony z opisem wydarzenia

Given: Użytkownik zalogowany
When: Klika link z nazwą wydarzenia
Then: Wyświetla się strona z opisem wydarzenia i jest przycisk, aby wziąć udział w wydarzeniu

• zrzuty ekranu poszczególnych części GUI z opisem workflow interfejsu graficznego
https://www.figma.com/file/gz1iazIktzwsQdHKMybPrj/Platform-for-organizing-events?type=design&node-id=0-1&mode=design&t=Tx9eDXNHxw4gbN1V-0

h) Opis infrastruktury sprzętowej / zasobów wykorzystywanych w projekcie

(powiązane z wykorzystywanymi technologiami)

Serwery i Infrastruktura Chmurowa:

Projekt wykorzystuje infrastrukturę chmurową, taką jak Microsoft Azure zapewniając elastyczność, skalowalność i dostępność.

Serwery w chmurze zapewniają niezbędną moc obliczeniową, pamięć oraz zasoby sieciowe do obsługi ruchu i przetwarzania danych.

Bazy Danych:

Do przechowywania danych aplikacji, używane jest Entity Framework wraz z bazą danych PostrgeSQL.

Bazy danych są optymalizowane pod kątem szybkiego dostępu do danych, a jednocześnie spełniają wymagania związane z integralnością i bezpieczeństwem danych.

ASP.NET Core Web API:

Warstwa backendowa aplikacji oparta jest na technologii ASP.NET Core Web API. Ten framework umożliwia łatwe tworzenie interfejsu programistycznego, obsługującego żądania HTTP i dostarczającego danych do interfejsu użytkownika Angular.

ASP.NET Core Web API jest zoptymalizowane pod kątem wydajności i skalowalności, co jest kluczowe w aplikacji obsługującej organizację wydarzeń.

Angular:

Warstwa front-endowa platformy opiera się na frameworku Angular. Angular dostarcza struktury do tworzenia dynamicznych i responsywnych interfejsów użytkownika.

Angular łączy się z ASP.NET Core Web API za pomocą zdefiniowanych interfejsów, pobierając i aktualizując dane bez konieczności przeładowywania całej strony, co wpływa pozytywnie na wydajność i responsywność interfejsu.

Angular Material:

Do zapewnienia jednolitego i atrakcyjnego wyglądu interfejsu użytkownika, projekt wykorzystuje Angular Material – zestaw gotowych komponentów, które są zgodne z zasadami Material Design.

Angular Material przyspiesza proces projektowania interfejsu, jednocześnie zapewniając konsystencję i estetykę.

i) Projekt makiet wizualnych (mockup) interfejsu graficznego GUI (o ile projekt taki

interfejs będzie posiadał; przydatne narzędzie: Figma)

https://www.figma.com/file/gz1iazIktzwsQdHKMybPrj/Platform-for-organizing-events?type=design&node-id=0-1&mode=design&t=gRDbz6M8pGyc6snX-0

j) Opis stosowanych środowisk w cyklu rozwoju projektu (uzupełnić)

Github – system kontroli wersji.

Jira – oprogramowanie firmy Atlassian służące do śledzenia błędów oraz zarządzania projektami.

GitHub Jira integration – rozszerzenie pokazuje zawartość połączonych zadań z Jira w Github

Neon – jest w pełni zarządzanym bezserwerowym Postgresem z hojnym wolnym poziomem. Neon oddziela pamięć masową i obliczeniową i oferuje nowoczesne funkcje programistyczne, takie jak bezserwerowe, rozgałęzione, bezdenne przechowywanie i inne.

Brevo Messaging API – API do wysyłki powiadomień przez mail (w darmowej wersji do 300 maili na dobę).

Azure Cloud – środowisko do rozmieszczenia i udostępnia projektów w chmurze.

Dokumentacja procesu Ci/CD i wyboru narzędzi

1. Wprowadzenie:

- **Cel:** Skrócenie cyklu życia dostarczania oprogramowania, zwiększenie stabilności i zapewnienie automatycznej dostarczalności aplikacji.
- **Korzyści:**
    - Szybsze wdrażanie zmian.
    - Większa pewność co do jakości kodu dzięki automatycznym testom.
    - Zautomatyzowane wdrażanie poprawek i nowych funkcji.

2. Konfiguracja Repozytorium Kodu:

- **Repozytorium:** Utwórz repozytorium kodu w Azure Repos lub podłącz istniejące repozytorium (np. GitHub, Bitbucket).
- **Gałęzie:** Ustal strategię gałęzi (np. **main** jako gałąź produkcyjna, **develop** dla wersji developerskiej).

3. Utworzenie Środowisk dla Testów:

- **Środowiska Testowe:** Skonfiguruj środowiska testowe, takie jak **Development**, **Staging** i **Production**.
- **Automatyczne Testy:** Skonfiguruj automatyczne testy jednostkowe, integracyjne i akceptacyjne.

4. Konfiguracja CI (Continuous Integration):

- **Build Pipeline:** Utwórz build pipeline w Azure DevOps.
    - Definiuj kroki budowania (np. kompilacja, testy jednostkowe).
    - Skonfiguruj zadania w przypadku sukcesu lub niepowodzenia.

5. Konfiguracja CD (Continuous Deployment):

- **Release Pipeline:** Utwórz release pipeline w Azure DevOps.
    - Skonfiguruj etapy dostarczania dla poszczególnych środowisk.
    - Zdefiniuj warunki automatycznego wdrożenia (np. po zakończeniu udanego buildu).

6. Wybór Narzędzi w Azure DevOps:

- **Azure Pipelines:** Narzędzie do konfiguracji CI/CD, oferujące elastyczność i integrację z wieloma platformami.
- **Azure Test Plans:** Umożliwia zarządzanie testami manualnymi i automatycznymi.
- **Azure Boards:** Narzędzie do śledzenia prac, zarządzania backlogiem i śledzenia postępów.

7. Integracja z Dodatkowymi Narzędziami:

- **Integracja z Aplikacjami Trzecimi:** Skonfiguruj integracje z narzędziami do monitorowania, raportowania błędów, itp.
- **Monitorowanie Działania Aplikacji:** Ustaw monitoring, aby śledzić wydajność i dostępność aplikacji.

8. Bezpieczeństwo:

- **Zabezpieczenia Dostępu:** Skonfiguruj odpowiednie uprawnienia dla zespołu projektowego.
- **Automatyczne Skanowanie Bezpieczeństwa:** Dodaj skanery bezpieczeństwa w procesie CI/CD.

9. Monitorowanie i Raportowanie:

- **Raportowanie Wyników Testów:** Skonfiguruj raporty na podstawie wyników testów.
- **Monitorowanie Procesu CI/CD:** Ustaw powiadomienia i monitoruj przebieg procesów CI/CD.

10. Dokumentacja:

- **Dokumentacja Procesu:** Przygotuj dokumentację procesu CI/CD, zawierającą instrukcje dotyczące konfiguracji, utrzymania i rozwiązywania problemów.

11. Skalowanie:

- **Automatyczne Skalowanie:** Skonfiguruj automatyczne skalowanie zasobów w chmurze w zależności od obciążenia aplikacji.

12. Płynne Wdrożenia:

- **Sloty dla Wdrożeń (Deployment Slots):** Używaj slotów wdrożeń do płynnych aktualizacji, minimalizując czas niedostępności.

13. Kontrola Wersji i Historii Wdrożeń:

- **Śledzenie Wersji:** Utrzymuj spójność wersji między środowiskami.
- **Historia Wdrożeń:** Zapewniaj pełną historię wdrożeń w celu łatwiejszego śledzenia zmian.