Uniwersytet Gdański Wydział Matematyki, Fizyki i Informatyki Instytut Informatyki

Oliver Gruba, Maciej Nasiadka

22 listopada 2025

Imię i Nazwisko (nr indeksu)	Oliver Gruba (292583) Maciej Nasiadka (292574)
Nazwa uczelni	Uniwersytet Gdański
Kierunek	Informatyka (profil praktyczny)
Prowadzący	dr inż. Stanisław Witkowski
Nazwa ćwiczenia	Modelowanie logiki kodu poprzez diagramy aktywności
Numer sprawozdania	5
Data zajęć	20.11.2025
Data oddania	26.11.2025
Miejscę na ocenę	

Spis treści

1. Określenie problemu projektowego.

1.1. Cel i zakres problemu projektowego

Celem projektu jest pokazanie, w jaki sposób diagramy aktywności mogą służyć do modelowania logiki działania aplikacji. Tego typu diagramy pozwalają na jednoznaczne, graficzne odwzorowanie przepływu sterowania, dzięki czemu ułatwiają analizę, implementację oraz późniejsze utrzymanie kodu. Problem projektowy polega więc na potrzebie przedstawienia logiki systemu w sposób klarowny, tak aby ograniczyć ryzyko błędnej interpretacji funkcjonalności przez programistów i projektantów.

1.2. Dlaczego właśnie diagram aktywności?

Diagram aktywności pozwala zaprezentować złożone operacje jako sekwencje kroków, ścieżek równoległych i warunkowych rozgałęzień. Dzięki temu stanowi narzędzie umożliwiające:

- uporządkowane przedstawienie logiki wykonania,
- zrozumienie zależności pomiędzy operacjami,
- identyfikację miejsc wymagających synchronizacji,
- ocenę, gdzie proces może zostać zrównoleglony.

Choć w literaturze określenia *diagram czynności* i *diagram aktywności* są zazwyczaj używane zamiennie, w tym projekcie skupiamy się na ich zastosowaniu do prezentowania **wewnętrznej logiki operacji systemu**, czyli modelowania tego, jak kod faktycznie działa.

1.3. Co powinien przedstawiać diagram aktywności?

Diagram aktywności opisuje m.in.:

- szczegółowe kroki realizacji funkcji systemu,
- przepływ sterowania między operacjami,
- warunki logiczne decydujące o wyborze kolejnych instrukcji,
- ścieżki wykonywane równolegle,
- podstawowy przepływ danych,

- sekwencję działań wykonywanych przez moduły systemu,
- zależności pomiędzy fragmentami implementacji.

Jego główną rolą jest przedstawienie, w jaki sposób system wykonuje daną procedurę.

1.4. Zastosowanie w projektowaniu logiki kodu

W kontekście projektowania i dokumentowania kodu diagramy aktywności są szczególnie przydatne, ponieważ umożliwiają:

- analizę operacji na poziomie wykonawczym,
- wykrywanie błędów sekwencji lub brakujących warunków,
- identyfikację kroków, które mogą być wykonywane współbieżnie,
- doprecyzowanie formalnego opisu procedur przed ich implementacją.

W efekcie stanowią one wsparcie zarówno na etapie programowania, jak i przy późniejszych modyfikacjach systemu.

1.5. Podsumowanie problemu i roli diagramu

Diagram aktywności pozwala skutecznie rozwiązać kluczowy problem projektowy: przedstawienie logiki systemu w sposób jednoznaczny i zrozumiały, z uwzględnieniem alternatywnych ścieżek wykonania, decyzji oraz zależności operacyjnych.

Jego wykorzystanie poprawia komunikację w zespole, ułatwia przyszłą rozbudowę projektu i pełni istotną rolę w dokumentacji technicznej, zwłaszcza gdy nad aplikacją pracuje wiele osób.

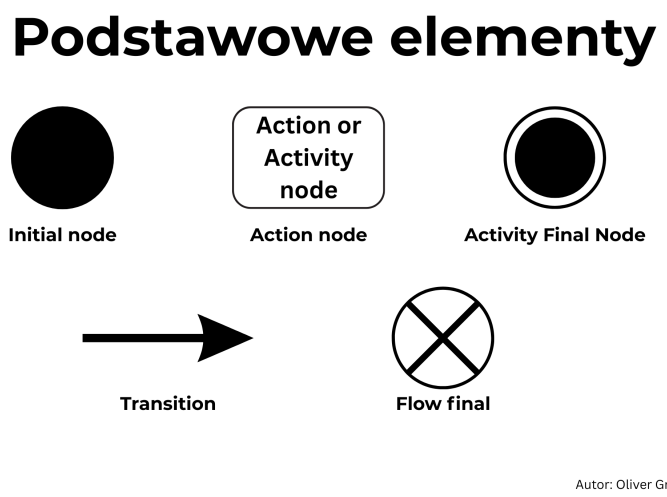
2. Notacje diagramu aktywności.

W diagramach aktywności wykorzystywane są standardowe elementy UML, które umożliwiają opis struktury przepływu sterowania i danych w procesach realizowanych przez system.:

2.1. Elementy podstawowe

- Początek (Initial Node) - punkt startowy procedury lub algorytmu. Zwykle jako czarne koło.
- Czynność (Action Node) - elementarna operacja wykonywana przez aplikację.

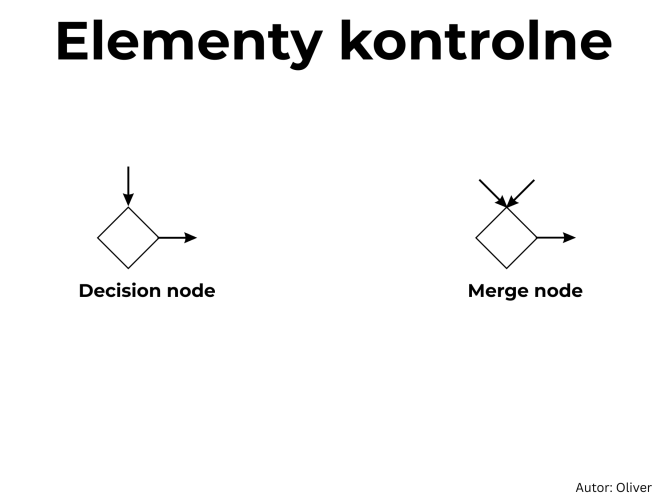
- Przejście (Transition) - kreśla przepływ sterowania między operacjami.
- Zakończenie (Activity Final Node) - zakończenie całego procesu wykonawczego.
- Flow Final - zakończenie pojedynczej ścieżki operacji.



Rysunek 1: Graficzne przedstawienie podstawowych elementów diagramu: Initial node, Action node, Transition, Activity final node i Flow final

2.2. Elementy kontrolne

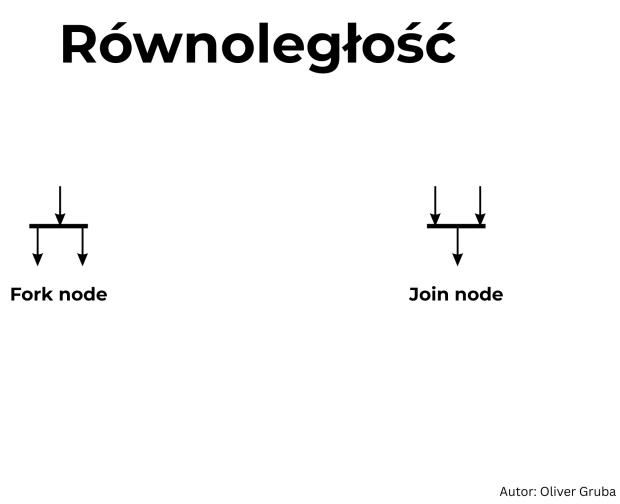
- Węzeł decyzyjny (Decision Node) - rozgałęzienie logiki operacji na podstawie warunku.
- Węzeł łączenia (Merge Node) - scalenie alternatywnych ścieżek sterowania.



Rysunek 2: Graficzne przedstawienie elementów kontrolnych diagramu: Decision node i Merge node

2.3. Równoległość

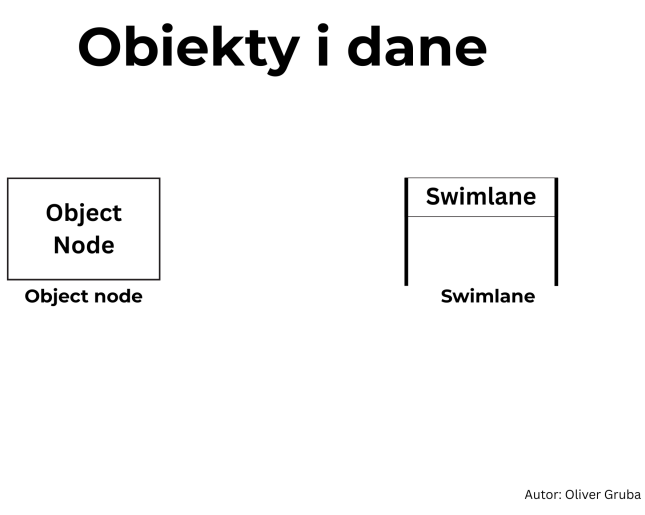
- Fork Node - rozdzielenie ścieżki sterowania na równoległe fragmenty wykonania.
- Join Node - synchronizacja kilku równoległych operacji.



Rysunek 3: Graficzne przedstawienie podstawowych elementów diagramu: Fork node i Join node

2.4. Obiekty i dane

- Object Node - przekazywane lub przetwarzane dane w trakcie operacji.
- Swimlane (tor aktywności) - sposób podziału odpowiedzialności między modułami systemu (np. Użytkownik, System, Moduł Powiadomień).



Rysunek 4: Graficzne przedstawienie podstawowych elementów diagramu: Object node i Swimlane

2.5. Semantyka

- Wykonanie operacji przebiega zgodnie z przepływem sterowania (Control Flow).
- Akcje traktowane są jako atomowe jednostki implementacji.
- Warunki muszą definiować jednoznaczny wybór ścieżki.
- Równoległość wymaga jawnej synchronizacji w celu zachowania spójności danych.

Elementy te łączy się za pomocą przepływów sterowania (*Control Flow*), które określają kolejność wykonywania czynności.

3. Budowa architektury realizacji usługi aplikacji

Tworzenie architektury realizacji usługi aplikacji z wykorzystaniem diagramów aktywności polega na odwzorowaniu logiki działania poszczególnych komponentów systemu oraz sposobu, w jaki współpracują one w trakcie wykonywania konkretnego zadania. Diagram aktywności stanowi w tym kontekście narzędzie umożliwiające analizę przepływu sterowania oraz danych, wspierając zarówno projektowanie usług, jak i ocenę ich spójności oraz efektywności.

3.1. Modelowanie komponentów i przepływu informacji

Z punktu widzenia architektury usługi kluczowe jest przedstawienie, w jaki sposób dane przepływają pomiędzy modułami aplikacji. Diagram aktywności pozwala uchwycić:

- przekazywanie danych pomiędzy elementami systemu,
- kierunek i kolejność komunikacji,
- punkty synchronizacji i rozdzielenia ścieżek operacji,
- kontekst danych niezbędny do wykonania każdej czynności.

Odwzorowanie przepływu informacji umożliwia ocenę, czy projektowana architektura minimalizuje redundancję, zapewnia odpowiednią separację odpowiedzialności oraz umożliwia późniejszą skalowalność systemu.

3.2. Reprezentacja kolejności operacji i logiki systemowej

Diagram aktywności przedstawia również logiczną strukturę działań składających się na daną usługę. Pozwala to określić:

- które operacje są wykonywane sekwencyjnie,
- w których punktach następują decyzje warunkowe,
- jakie działania mogą być wykonywane współbieżnie,
- jak przebiega przepływ sterowania po zakończeniu danej czynności.

Precyzyjne rozpisanie kolejności operacji jest kluczowe przy analizie wydajności systemu oraz identyfikacji kroków podatnych na błędy lub wymagających optymalizacji.

3.3. Obsługa wyjątków i przypadków brzegowych

W projektowaniu architektury usługi konieczne jest uwzględnienie scenariuszy niepoprawnych lub nietypowych. Diagram aktywności pozwala wyraźnie wskazać:

- momenty wykrywania błędów,
- alternatywne ścieżki wykonania,
- procesy uzupełniające (np. rejestracja błędów, komunikaty zwrotne),
- zasoby wymagające wycofania lub ponownej alokacji.

Dzięki temu architektura usługi uwzględnia mechanizmy odporności na błędy i pozwala ocenić jakość obsługi sytuacji wyjątkowych.

3.4. Integracja modułów i zależności pomiędzy elementami aplikacji

Diagram aktywności umożliwia identyfikację zależności pomiędzy elementami implementacji, takich jak:

- kolejność wywołań modułów,
- wymogi dotyczące stanu systemu przed wykonaniem operacji,
- zależności danych w kontekście kolejnych działań,
- powiązania logiczne wpływające na działanie całej usługi.

W ten sposób diagram aktywności wspiera tworzenie architektury usług, które są spójne, odporne na błędy i zgodne z założeniami projektowymi.

4. Procedura wykonania operacji informatycznej

Procedura realizacji operacji informatycznej przedstawiona poprzez diagram aktywności pozwala uchwycić szczegółową logikę wykonania zadania, od momentu inicjalizacji po zakończenie działania systemu. Dzięki takiemu modelowi możliwe jest zarówno zrozumienie przebiegu operacji z perspektywy użytkownika, jak i uwzględnienie wewnętrznych kroków realizowanych przez system.

4.1. Inicjalizacja działania i odbiór żądania

Proces rozpoczyna się od zainicjowania operacji, najczęściej poprzez:

- wysłanie żądania przez użytkownika,
- wywołanie usługi przez inny moduł systemu,
- zewnętrzny sygnał lub zdarzenie (np. czasowe, systemowe).

Etap ten stanowi punkt wejścia do procedury i określa kontekst wykonywanych dalej działań.

4.2. Weryfikacja danych wejściowych i warunków początkowych

Kolejnym krokiem jest sprawdzenie, czy dane wejściowe są poprawne oraz czy system znajduje się w stanie umożliwiającym wykonanie operacji. Proces ten może obejmować:

- walidację wartości dostarczonych przez użytkownika,
- kontrolę integralności danych,
- sprawdzenie uprawnień,
- weryfikację dostępności wymaganych zasobów.

W przypadku niepowodzenia system może przejść do ścieżki obsługi błędu lub alternatywnego scenariusza.

4.3. Podejmowanie decyzji i wybór ścieżki wykonania

Po weryfikacji danych system dokonuje wyboru dalszej ścieżki na podstawie zdefiniowanych warunków logicznych. Obejmuje to:

- analizę spełnienia warunków decyzyjnych,

- wybór odpowiednich operacji lub procedur,
- inicjację procesów równoległych, jeśli sytuacja tego wymaga.

Warunki te determinują, w jaki sposób operacja będzie przebiegać dalej.

4.4. Realizacja operacji systemowych

Właściwe wykonanie procedury polega na realizacji zdefiniowanych działań. Mogą one obejmować:

- przetwarzanie danych w modułach aplikacji,
- wykonywanie operacji równoległych,
- komunikację z bazą danych lub usługami zewnętrznymi,
- wywoływanie dodatkowych procesów pomocniczych.

Diagram aktywności pozwala przy tym jasno wskazać punkty synchronizacji oraz miejsca potencjalnych konfliktów lub blokad.

4.5. Zapis wyników i zwrócenie odpowiedzi

Po zakończeniu właściwego przetwarzania system:

- zapisuje dane w repozytorium,
- aktualizuje stan zasobów,
- generuje wynik operacji i przekazuje go do użytkownika lub innego modułu.

Ten etap jest kluczowy dla zapewnienia integralności danych oraz spójności stanu systemu.

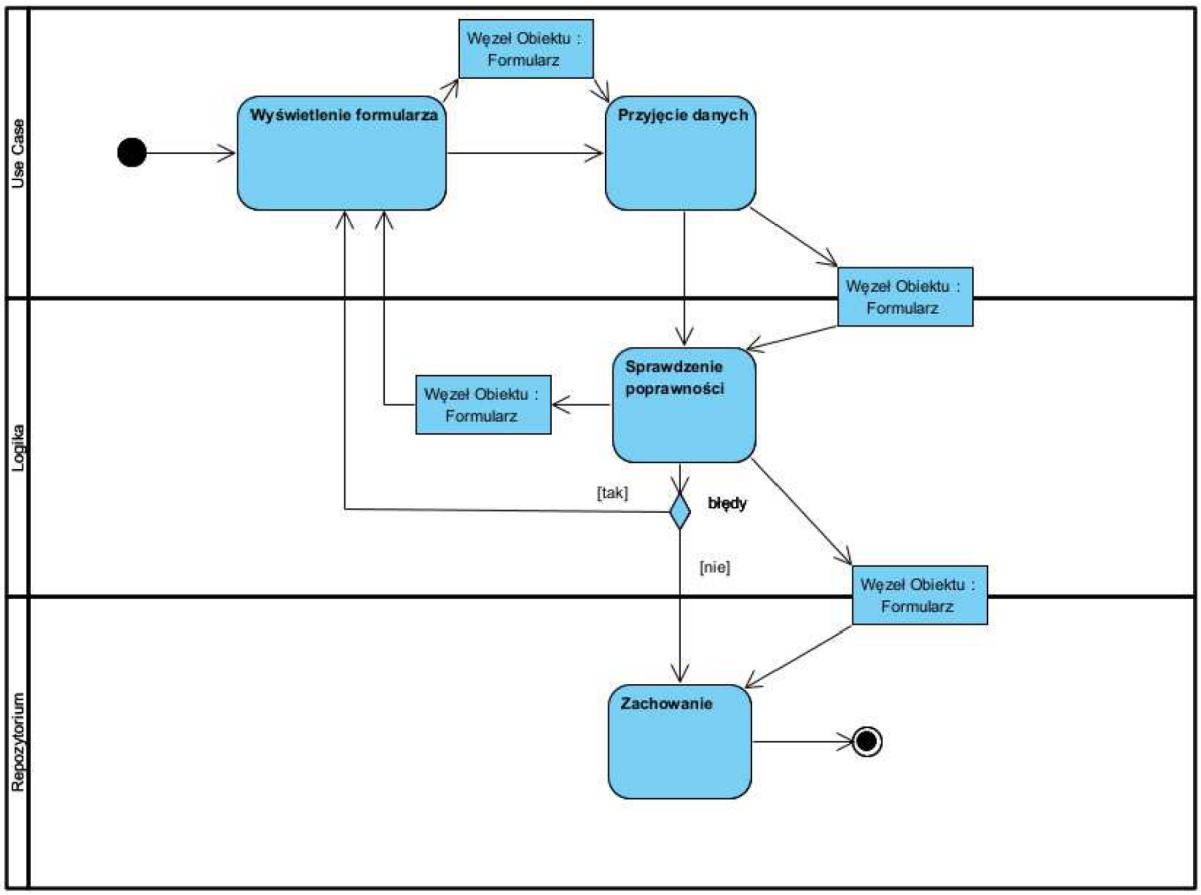
4.6. Zakończenie procesu i zwolnienie zasobów

Proces kończy się zwolnieniem zasobów, takimi jak połączenia sieciowe, uchwyty plików czy tymczasowe fragmenty pamięci. W niektórych implementacjach etap ten obejmuje również:

- rejestrowanie informacji diagnostycznych,
- aktualizację logów systemowych,
- zwolnienie blokad wykorzystywanych przez procedurę.

Zamknięcie operacji zapewnia poprawne funkcjonowanie systemu i minimalizuje ryzyko wycieków zasobów.

5. Zadanie 1 - diagram aktywności według pliku od prowadzącego



Rysunek 5: Diagram aktywności przedstawiający czynności związane z wypełnianiem formularza z podziałem na 3 swimlanes: use cases, logika i repozytorium

Poniższy diagram aktywności przedstawia podstawowy proces obsługi formularza w systemie: od momentu jego wyświetlenia, przez odbiór danych i walidację, aż do podjęcia decyzji o zakończeniu procedury lub ponownym zwróceniu formularza z błędami. Diagram został podzielony na trzy torowiska (swimlanes): Use Case, Logika, Repozytorium, co pozwala odróżnić działania użytkownika, procedury przetwarzania oraz operacje końcowe.

5.1. Struktura diagramu i przepływ sterowania

1. Inicjalizacja procesu (Use Case) Proces rozpoczyna się w torze **Use Case**. Pierwszą aktywnością jest:

    - Wyświetlenie formularza - system prezentuje użytkownikowi interfejs wejściowy.

Węzeł obiektu “Formularz” reprezentuje przekazywaną strukturę danych.

Po prezentacji formularza system oczekuje na interakcję użytkownika.

2. Przyjęcie danych od użytkownika (Use Case -> Logika)

   Kolejna aktywność to:

   - Przyjęcie danych - formularz zostaje odesłany do systemu wraz z wartościami pól.

     W tym miejscu dane są przekazywane do toru logicznego.

   Dołączony jest drugi węzeł obiektu “Formularz”, który obrazuje, że system pracuje już na strukturze wypełnionej przez użytkownika.

3. Sprawdzanie poprawności (Logika)

   Najważniejsza część procesu:

   - Sprawdzenie poprawności - system przeprowadza walidację na podstawie reguł logicznych.

   Walidacja prowadzi do podjęcia decyzji:

   - [**tak**] błędy - jeżeli dane są niepoprawne.
   - [**nie**] - jeżeli dane przeszły walidację.

   W przypadku wykrycia błędów system:

   - zwraca formularz do wcześniejszego etapu (pętla odzwierciedlona strzałkami),
   - umożliwia ponowną edycję danych przez użytkownika.

   Całość odzwierciedla klasyczny wzorzec: validate –> return –> correct –> resubmit

4. Przekazanie procesu do Repozytorium

   Gdy dane są poprawne, przepływ przechodzi do toru **Repozytorium**:

   - Węzeł obiektu ”Formularz” reprezentuje przekazywaną, zwalidowaną strukturę danych,
   - Następnie wykonywana jest aktywność końcowa:
     - Zachowanie - operacja systemowa, np. zapis danych, wygenerowanie rekordu, wykonanie dalszej logiki.

   Po wykonaniu tej operacji proces zostaje zakończony.

5.2. Interpretacja logiki działania systemu

Diagram ukazuje logiczny i uporządkowany przepływ informacji pomiędzy trzema warstwami:

1. Warstwa przypadków użycia (Use Case)
    - Interakcje użytkownika.
    - System wyświetla formularz i odbiera dane wejściowe.
2. Warstwa logiki biznesowej (Logika)
    - Realizowane są tu wewnętrzne algorytmy walidacyjne.
    - Gałęzie decyzyjne obsługują błędy i zapewniają pętlę poprawiania danych.
    - Warstwa ta odpowiada za poprawność i spójność danych przed jakąkolwiek ich persystencją.
3. Warstwa repozytorium (Repozytorium)
    - Wykonywane są operacje trwałe: zapis, aktualizacja, dalsze procesy systemowe.
    - Proces kończy się w tym torze, co podkreśla, że finalny rezultat jest wynikiem poprawnie zweryfikowanych danych.

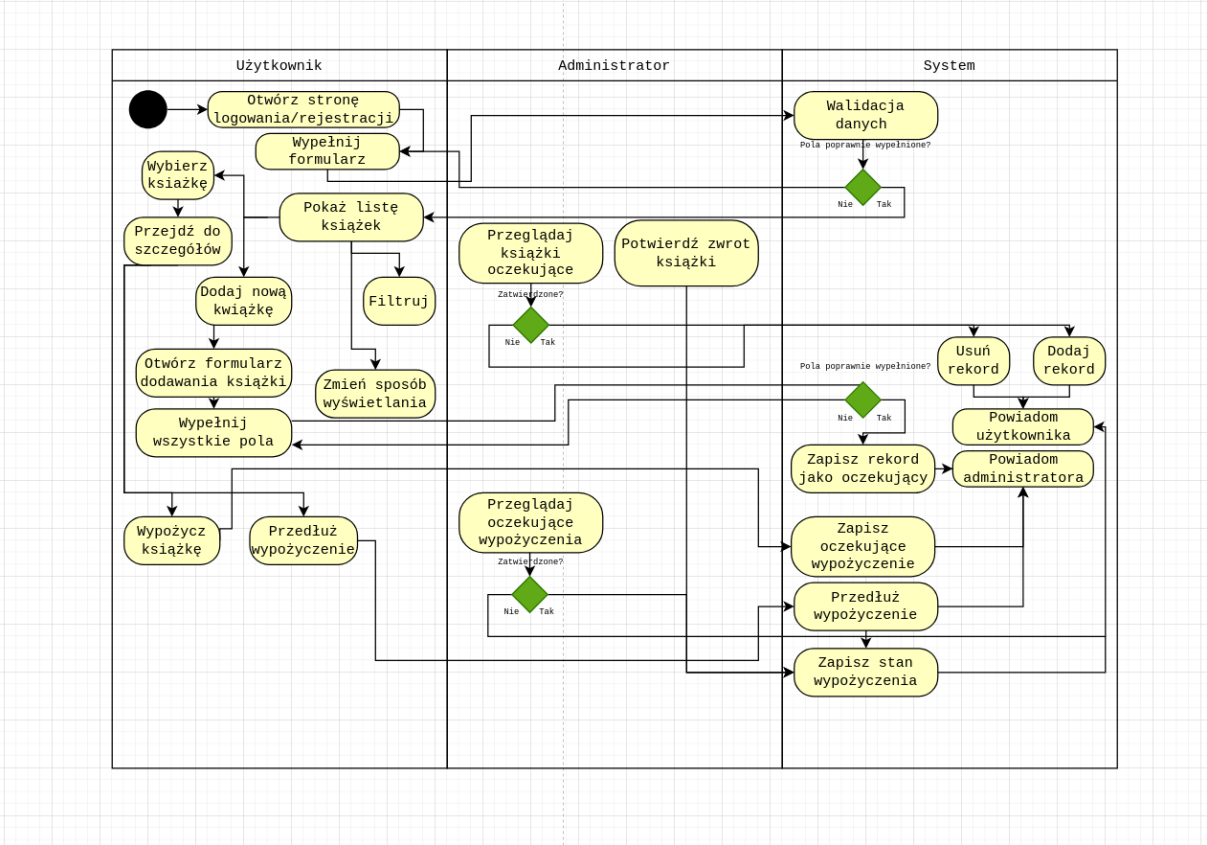
5.3. Podsumowanie działania procesu

Diagram przedstawia klasyczny scenariusz obsługi formularza w aplikacji:

1. Prezentacja –> Odbiór –> Walidacja –> Zapis
2. Obsługa błędów odbywa się w formie powrotnej pętli.
3. Przepływ między torami jasno rozdziela odpowiedzialności warstw systemu.

Układ diagramu jest czytelny i zgodny z praktyką modelowania przepływu działań w systemach wymagających walidacji wejścia oraz bezpiecznego przetwarzania danych.

6. Zadanie 2 - diagram aktywności według systemu czytelni



Rysunek 6: Diagram aktywności przedstawiający działanie systemu bibliotecznego

Diagram aktywności prezentuje logikę operacji wykonywanych w systemie bibliotecznym. Obejmuje zarówno procesy związane z obsługą użytkownika, jak i wewnętrzne działania systemu odpowiedzialne za zarządzanie danymi i stanem wypożyczeń.

6.1. Proces uwierzytelniania użytkownika

Pierwszym etapem jest inicjalizacja aplikacji poprzez formularz logowania lub rejestracji. Diagram obejmuje:

- walidację danych wejściowych,
- obsługę błędów uwierzytelniania,
- przekierowanie użytkownika do interfejsu głównego po poprawnym logowaniu.

Proces ten reprezentuje moduł bezpieczeństwa odpowiadający za dostęp do systemu.

6.2. Przeglądanie zasobów bibliotecznych

Po zalogowaniu użytkownik przechodzi do modułu katalogu. Diagram przedstawia:

- wyświetlenie listy dostępnych pozycji,
- filtrowanie i wyszukiwanie zasobów,
- pobieranie szczegółów wybranej książki.

Sekcja ta odwzorowuje komunikację między warstwą prezentacji a modułem zarządzania katalogiem.

6.3. Obsługa operacji wypożyczeń

Diagram zawiera trzy główne ścieżki, które system może wykonać:

- wypożyczenie książki,
- przedłużenie trwającego wypożyczenia,
- zwrot egzemplarza.

Każda ścieżka zawiera etap walidacji, aktualizacji bazy danych oraz generowania powiadomień. Alternatywne przejścia są odwzorowane poprzez rozgałęzienia decyzyjne.

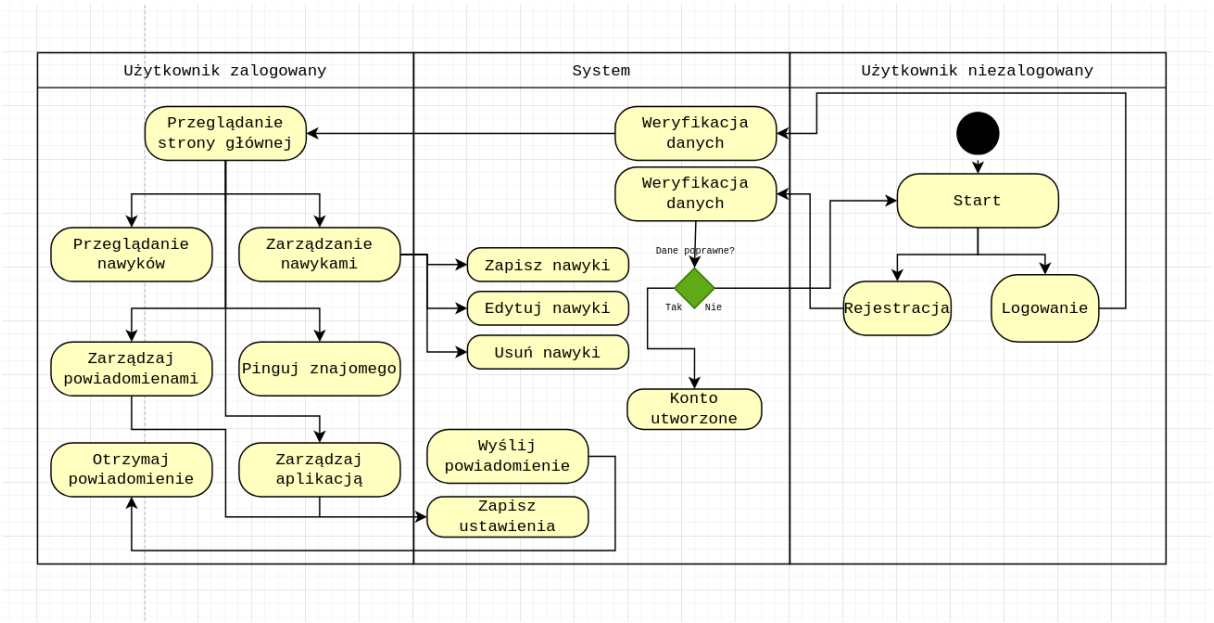
6.4. Zapis i powiadomienia systemowe

Końcowa część procesu obejmuje operacje systemowe:

- zapis stanu wypożyczeń,
- aktualizację historii użytkownika,

wysłanie komunikatu o powodzeniu lub błędzie. Te czynności stanowią końcowy etap przepływu logiki.

7. Zadanie 3 - diagram aktywności według projektu zespołowego



Rysunek 7: Diagram aktywności Kairo Habit App

Diagram aktywności przedstawia szczegółową logikę działania aplikacji *Kairo Habit App*, odwzorowując sposób, w jaki system przetwarza dane użytkownika oraz realizuje operacje związane z zarządzaniem nawykami.

7.1. Rejestracja i logowanie użytkownika

Model obejmuje dwa warianty wejścia do aplikacji:

- pełną rejestrację konta wraz z walidacją danych: weryfikacja nazwy użytkownika, struktury hasła oraz adresu e-mail,
- logowanie z wykorzystaniem wcześniej zapisanych danych i tokenów sesyjnych.

Proces uwzględnia również obsługę błędów i ścieżkę alternatywną dla niepoprawnych danych.

7.2. Walidacja i przetwarzanie danych użytkownika

W diagramie przedstawiono logikę związaną z obsługą danych użytkownika, obejmującą:

- sprawdzanie ważności tokenu sesji,

- walidację danych pobieranych z pamięci lokalnej lub bazy,
- aktualizację konfiguracji użytkownika przy starcie aplikacji.

Sekcja ta odzwierciedla współpracę modułów odpowiedzialnych za bezpieczeństwo, sesje oraz repozytorium danych.

7.3. Obsługa modułów funkcjonalnych aplikacji

Diagram uwzględnia główne moduły aplikacji:

1. moduł nawyków:
    - tworzenie wpisów,
    - edytowanie wpisów,
    - usuwanie wpisów.
2. moduł powiadomień:
    - harmonogramy.
    - synchronizacja,
    - obsługa zdarzeń.
3. moduł ustawień:
    - konfiguracja preferencji użytkownika.

Każdy moduł operuje na współdzielonych zasobach i posiada własne warunki wykonania, przedstawione jako rozgałęzienia decyzyjne.

7.4. Zapis, edycja i synchronizacja danych

Ostatni etap obejmuje:

- zapis zmian w pamięci lokalnej i bazie danych,
- synchronizację z serwerem,
- aktualizację stanu aplikacji po każdej operacji,
- obsługę równoległych procesów synchronizacyjnych.

Sekcja ta odwzorowuje implementacyjną logikę współpracy modułów z elementami infrastruktury aplikacji (API, repozytorium, warstwa synchronizacji).

8. Zastosowania diagramów aktywności

Diagramy aktywności znajdują zastosowanie wszędzie tam, gdzie konieczne jest precyzyjne odwzorowanie logiki operacji systemowych:

8.1. Projektowanie logiki aplikacji

- Umożliwiają analizę przepływu danych i sterowania.
- Pozwalają projektantom zaplanować implementację modułów.

8.2. Modelowanie procedur i algorytmów

- Pomagają w formalnym opisaniu procesów wykonywanych w kodzie.
- Stanowią uzupełnienie pseudokodu i diagramów sekwencji.

8.3. Dokumentacja implementacji

- Wspierają zrozumienie kodu przez cały zespół developerski.
- Ułatwiają utrzymanie i rozwijanie aplikacji.

8.4. Testowanie i weryfikacja logiki

- Służą za podstawę do projektowania testów integracyjnych i systemowych.
- Ułatwiają wykrywanie ścieżek alternatywnych i wyjątków.

8.5. Optymalizacja implementacji

- Pozwalają ocenić miejsca do równoległego wykonywania operacji,
- ujawniają potencjalne wąskie gardła procedur.

9. Wnioski

Wnioski z analizy oraz tworzenia diagramów aktywności można podzielić na trzy główne obszary: znaczenie modelowania, wpływ na projektowanie architektury oraz zastosowanie w praktycznej realizacji projektu zespołowego.

9.1. Znaczenie modelowania procesów

Diagramy aktywności stanowią jedno z kluczowych narzędzi do odwzorowania logiki wykonawczej systemu. Dzięki nim możliwe jest:

- szczegółowe prześledzenie kolejnych etapów realizacji operacji,
- precyzyjne określenie warunków decyzyjnych i gałęzi alternatywnych,
- identyfikacja błędów w przepływach procesów jeszcze przed etapem implementacji,
- zweryfikowanie czy podział logiki na moduły jest spójny i kompletny.

Modelowanie w ten sposób pozwala lepiej zrozumieć wewnętrzne działanie systemu oraz ocenić, czy procesy użytkownika zostały odwzorowane zgodnie z założeniami funkcjonalnymi.

9.2. Wpływ na projektowanie architektury

W kontekście tworzenia architektury aplikacji diagramy aktywności pełnią funkcję narzędzia porządkującego:

- ujawniają przepływ informacji pomiędzy komponentami aplikacji,
- ułatwiają identyfikację zależności pomiędzy modułami,
- pozwalają analizować scenariusze wyjątków oraz sposób obsługi błędów,
- umożliwiają wczesną ocenę wydajnościową (np. wykrycie potencjalnych blokad i nieefektywnych ścieżek).

Dzięki takiemu podejściu projekt architektury staje się bardziej kompletny, a dokumentacja logiczna wyraźnie wspiera proces implementacji.

9.3. Zastosowanie w projekcie zespołowym

W przypadku złożonych projektów, takich jak aplikacja *Kairo Habit App*, diagramy aktywności odgrywają rolę narzędzia koordynacji:

- zapewniają wspólne zrozumienie sposobu działania modułów przez wszystkich członków zespołu,
- porządkują proces implementacji i wyznaczają granice odpowiedzialności,
- upraszczają planowanie prac nad integracją komponentów,

- stanowią podstawę do projektowania testów jednostkowych i integracyjnych.

Tego typu modelowanie zmniejsza ryzyko niespójności między poszczególnymi częściami projektu i ułatwia utrzymanie jednolitego kierunku implementacyjnego. Trzeba jednak pamiętać, że nie ważne jak dobrze udokumentujemy całość, niezrozumienie wśród członków zespołu do pewnego stopnia jest normalną częścią życia i pracy jako informatyk lub programista.

9.4. Podsumowanie

Ostatecznie diagramy aktywności pełnią rolę połączenia pomiędzy analizą funkcjonalną a implementacją. Umożliwiają jasne odwzorowanie operacji aplikacji, wspierają poprawność projektowania i przyczyniają się do zwiększenia jakości kodu. Dzięki temu stanowią jedno z podstawowych narzędzi w procesie inżynierii oprogramowania, szczególnie w środowiskach zespołowych; jako narzędzie do zmniejszenia ilości nieporozumień w zespole poprzez ustandaryzowany sposób przekazania wiedzy jak działające moduły aplikacji.