Uniwersytet Gdański Wydział Matematyki, Fizyki i Informatyki Instytut Informatyki

Oliver Gruba, Maciej Nasiadka

23 grudnia 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	Interakcja procesu programu informatycznego w czasie poprzez tworzenie diagramu sekwencji
Numer sprawozdania	6
Data zajęć	27.11.2025
Data oddania	03.12.2025
Miejscę na ocenę	

Spis treści

1. Procedura uruchamiania diagramu sekwencji

Diagram sekwencji jest narzędziem dynamicznym, służącym do odwzorowania zachowania systemu w czasie. Aby poprawnie zinterpretować, uruchomić i przeanalizować diagram sekwencji, konieczne jest zastosowanie określonej procedury. Pozwala ona na konsekwentne prześledzenie przepływu komunikatów pomiędzy aktorami i obiektami, a także na analizę logiki procesów biznesowych.

1.1. Identyfikacja aktorów i obiektów uczestniczących w procesie

- Określenie, które elementy systemu biorą udział w analizowanej interakcji.
- Wyróżnienie aktorów zewnętrznych (np. użytkownik, system nadrzędny).
- Wyodrębnienie obiektów wewnętrznych systemu (np. podsystemy).
- Ustalenie zależności między elementami oraz zakresu ich odpowiedzialności.
  Np. rozróżnienie między komunikacją między frontendem, backendem i bazą danych.

1.2. Określenie scenariusza bazowego

- Zdefiniowanie kolejnych kroków przewidywanej interakcji.
- Zapisanie scenariusza w sposób sekwencyjny, od pierwszego do ostatniego kroku.
  Gdzie idąc kolejno w dół diagrmau pokazujemy elementy, które wykonują się później w systemie (zachowując hierarchie kolejności procesów).
- Uwzględnienie logiki biznesowej bez wprowadzania wyjątków.

1.3. Dodanie scenariuszy alternatywnych

- Identyfikacja możliwych punktów decyzyjnych w procesie.
- Określenie sytuacji wyjątkowych (np. błędy, odmowy dostępu, brak środków).
- Uporządkowanie alternatyw przy użyciu konstrukcji `alt` i `opt`.
- Zapewnienie kompletności scenariuszy poprzez uwzględnienie wszystkich możliwych ścieżek.

1.4. Modelowanie komunikacji między obiektami

- Przedstawienie wywołań metod, wymiany komunikatów, sygnałów asynchronicznych.
- Określenie, czy komunikacja jest synchroniczna (blokująca), czy asynchroniczna.
- Uporządkowanie komunikatów od góry do dołu, zgodnie z czasem ich wystąpienia.

1.5. Analiza i walidacja poprawności diagramu

- Sprawdzenie spójności diagramu z rzeczywistym przebiegiem operacji.
- Weryfikacja kompletności procesu oraz zgodności ze specyfikacją wymagań.
- Ocena poprawności połączeń pomiędzy liniami życia.
- Eliminacja błędów logicznych oraz niepotrzebnych elementów.

2. Cel tworzenia diagramu sekwencji

Diagram sekwencji jest kluczowym narzędziem analizy dynamicznej systemu, pozwalającym na precyzyjne odwzorowanie logiki przepływu zdarzeń w czasie. Jego głównym celem jest wyjaśnienie sposobu współpracy obiektów i aktorów podczas realizacji konkretnej funkcji systemu.

2.1. Przedstawienie dynamiki systemu w czasie

- Umożliwia modelowanie zachowania systemu w odpowiedzi na działania aktora.
- Ukazuje kolejność operacji oraz reakcje obiektów.
- Pozwala obserwować logikę sterowania procesem w sposób chronologiczny.

2.2. Dokumentacja interakcji między obiektami

- Wskazuje sposób komunikacji pomiędzy komponentami systemu.
- Wyjaśnia, jakie metody lub komunikaty są wywoływane.
- Umożliwia zrozumienie odpowiedzialności poszczególnych elementów systemu.

2.3. Wspomaganie projektowania architektury

- Ułatwia podział systemu na moduły oraz określenie ich odpowiedzialności.
- Pomaga wykryć przeciążone lub zbędne elementy logiki.
- Przedstawia miejsca, w których występuje zależność czasowa.

2.4. Analiza warunków wyjątkowych i przypadków alternatywnych

- Pozwala wykryć potencjalne błędy na wczesnym etapie projektowania.
- Ułatwia dokumentowanie sytuacji, w których logika procesu się rozgałęzia.
- Formalizuje decyzje i punkty kontrolne w systemie.

2.5. Wspieranie komunikacji w zespole projektowym

- Stanowi wspólny język pomiędzy analitykami, programistami i testerami.
- Ujednolica sposób interpretacji funkcjonalności systemu.
- Jest przydatny w trakcie przeglądów technicznych oraz analiz kodu.

3. Linie życia aktorów i obiektów

Linie życia (*lifelines*) stanowią podstawowy element diagramu sekwencji i służą do reprezentacji istnienia aktora lub obiektu w czasie. Każdy uczestnik interakcji posiada własną linię życia, która biegnie pionowo od góry diagramu w dół.

3.1. Definicja linii życia

- Jest to pionowa linia symbolizująca czas istnienia obiektu w trakcie wykonywania procesu.
- Linia zaczyna się w momencie, w którym obiekt pojawia się w systemie, a kończy w chwili zakończenia interakcji.
- Może reprezentować zarówno aktorów zewnętrznych, jak i obiekty systemowe.

3.2. Reprezentacja aktywności obiektu

- Aktywność często oznaczana jest prostokątnym paskiem (tzw. *activation bar*).
- Pasek aktywności pokazuje okres, w którym obiekt wykonuje operację lub przetwarza dane.
- Dłuższe okresy aktywności mogą oznaczać intensywną logikę obliczeniową lub oczekiwanie na odpowiedź.

3.3. Komunikaty wpływające na linię życia

- Komunikaty synchroniczne rozpoczynają aktywność obiektu (np. wywołanie metody).
- Komunikaty zwrotne kończą aktywność lub sygnalizują rezultat operacji.
- Komunikaty asynchroniczne mogą nie wymagać aktywności zwrotnej.

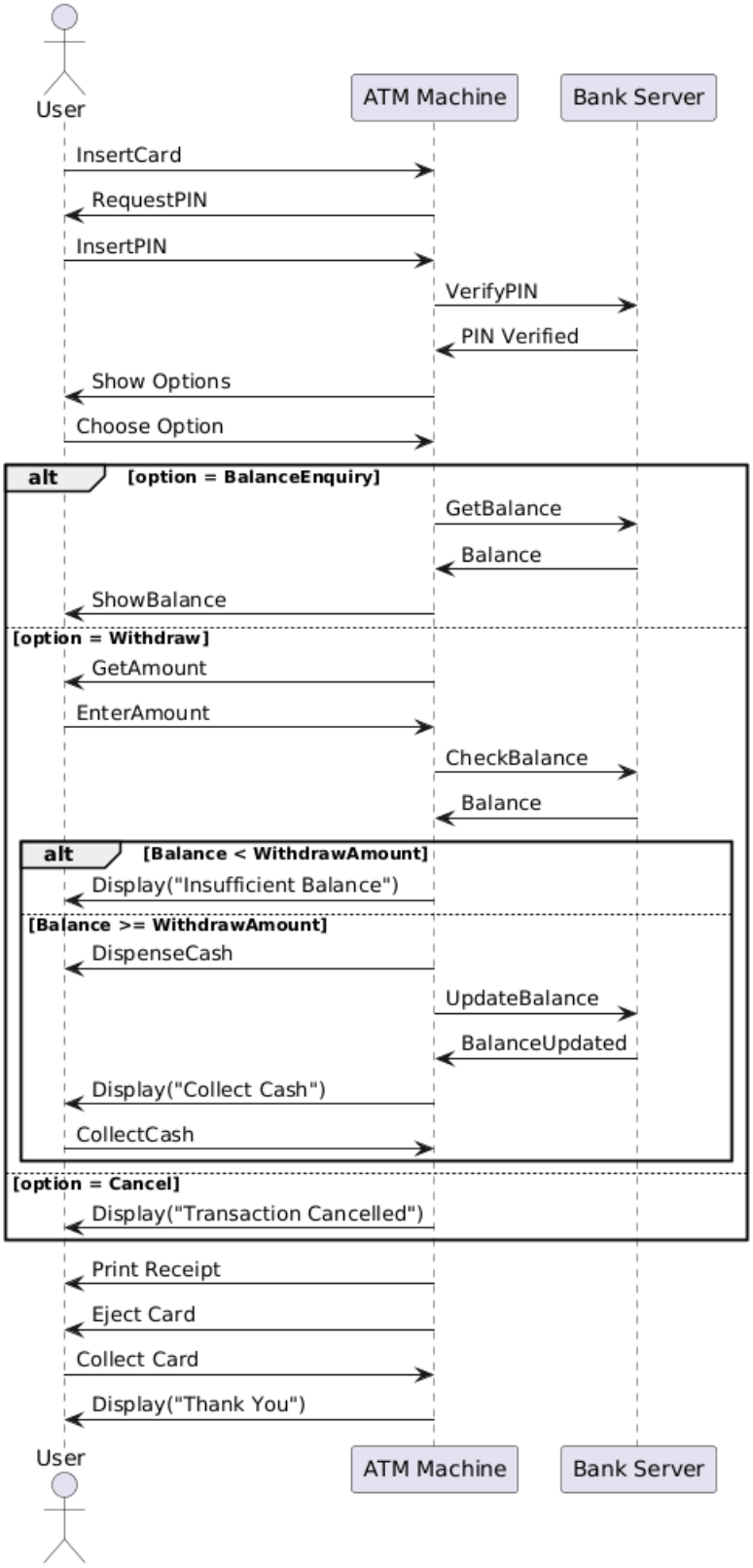
3.4. Zależności czasowe

- Im niżej umieszczony komunikat, tym później występuje w czasie.
- Linie życia pozwalają analizować, jak długo trwa przetwarzanie poszczególnych operacji.
- Dzięki nim możliwe jest wskazanie wąskich gardeł lub punktów blokujących system.

3.5. Interpretacja zakończenia życia obiektu

- Zakończenie linii życia często przedstawione jest za pomocą symbolu X.
- Oznacza moment, w którym obiekt jest usuwany lub kończy działanie w procesie.
- W przypadku aktorów zakończenie linii życia sygnalizuje zakończenie interakcji użytkownika z systemem.

4. Zadanie 1 - przykład z materiałów.



Rysunek 1: Diagram sekwencji z materiałów od prowadzącego. Przedstawiający użycie bankomatu przez użytkownika i przebieg aktualizacji systemu serwera bankowego.

4.1. Rozpoczęcie interakcji: uwierzytelnienie użytkownika

Proces rozpoczyna się, gdy Użytkownik umieszcza kartę w bankomacie:

- **Użytkownik** wykonuje akcję *Insert Card*.
- **ATM** odpowiada żądaniem wprowadzenia numeru PIN *Request PIN*.
- **Użytkownik** wprowadza PIN, wysyłając komunikat *Insert PIN* do **ATM**.
- **ATM** przekazuje PIN do **Systemu Bankowego** poprzez komunikat *Verify PIN*.
- **System Bankowy** odpowiada komunikatem *PIN Verified*, potwierdzając poprawność danych uwierzytelniających.

Jeżeli uwierzytelnienie przebiegnie pomyślnie, użytkownik może kontynuować proces.

4.2. Wybór operacji przez użytkownika

Po poprawnej weryfikacji PIN **ATM**:

- wyświetla możliwe opcje operacji *Show Options*,
- użytkownik wybiera jedną z nich *Choose Option*.

Dalszy przebieg procesu zależy od wybranego wariantu, co przedstawione jest na diagramie za pomocą konstrukcji alternatywnej `alt`.

4.3. Opcja: Sprawdzenie salda

Jeżeli **Użytkownik** wybiera *Balance Enquiry*, wykonywane są następujące kroki:

- **ATM** wysyła do Systemu Bankowego żądanie *Get Balance*.
- **System Bankowy** zwraca bieżące saldo konta (*Balance*).
- **ATM** wyświetla saldo użytkownikowi poprzez komunikat *Show Balance*.

Proces tej opcji kończy się na wyświetleniu informacji.

4.4. Opcja: Wypłata gotówki

Jeżeli **Użytkownik** wybiera opcję *Withdraw*, proces przebiega dalej:

- **ATM** prosi użytkownika o podanie kwoty do wypłaty *Get Amount*.
- **Użytkownik** wprowadza kwotę *Enter Amount*.

- **ATM** przesyła żądanie do Systemu Bankowego: *Check Balance*.
- **System Bankowy** zwraca aktualne saldo (*Balance*).

W tym momencie pojawia się alternatywa:

1. **Saldo niewystarczające**
   Jeżeli saldo jest mniejsze niż żądana kwota:
   - **ATM** wyświetla komunikat *Insufficient Balance*.
2. **Saldo wystarczające**
   Jeżeli saldo jest wystarczające:
   - **ATM** uruchamia wypłatę gotówki *Dispense Cash*.
   - **System Bankowy** aktualizuje saldo konta (*Update Balance*).
   - Po zakończeniu operacji **Użytkownik** otrzymuje komunikat: *Collect Cash* i odbiera gotówkę (*Collect Cash*).

4.5. Opcja: Anulowanie transakcji

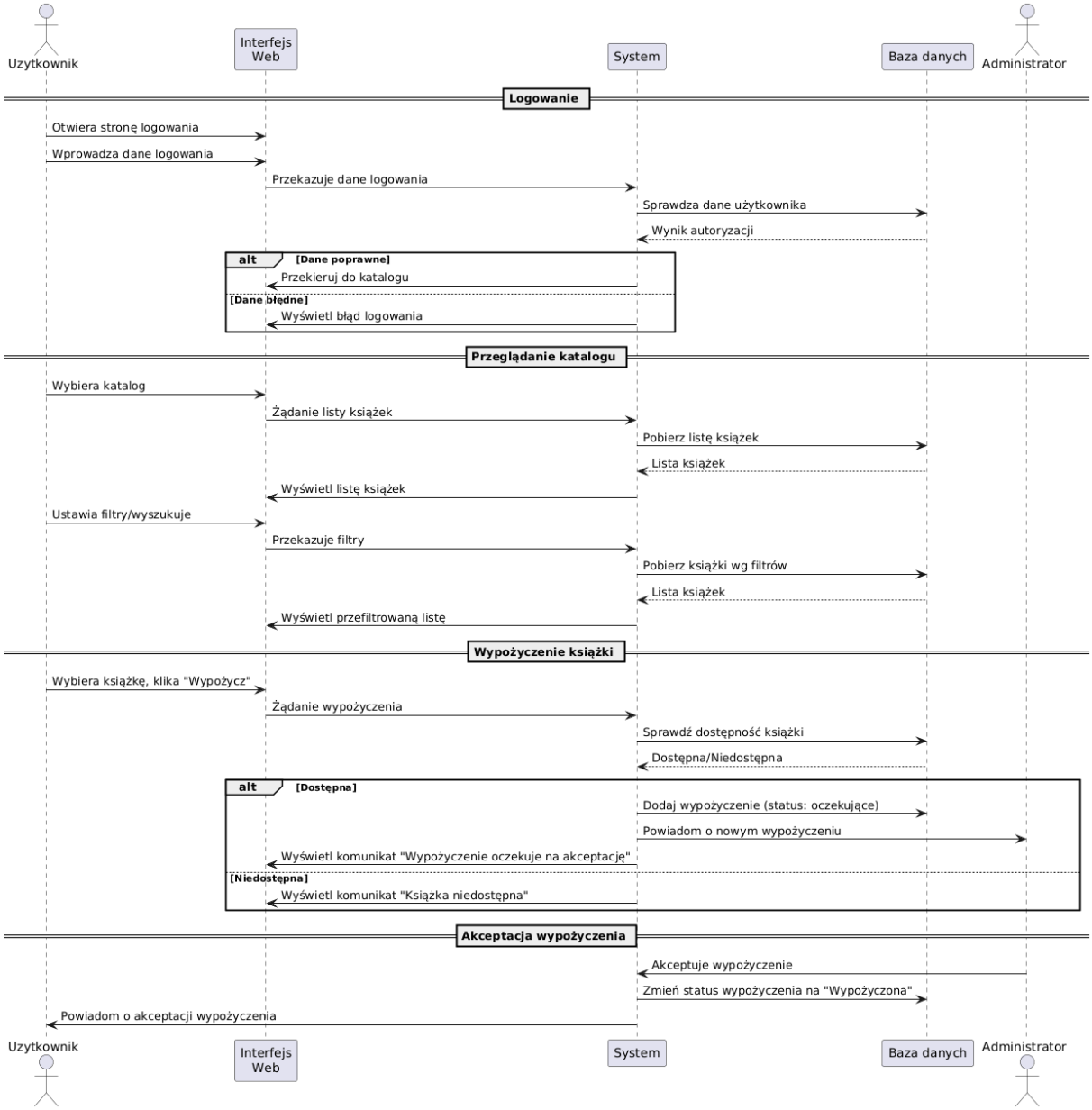
Jeżeli użytkownik wybierze opcję *Cancel*, ATM wyświetla komunikat *Transaction Cancelled*. Proces wypłaty lub sprawdzania salda zostaje zakończony.

4.6. Zakończenie procesu

W końcowej części, niezależnie od wcześniejszej ścieżki:

- **ATM** drukuje potwierdzenie operacji *Print Receipt*.
- **ATM** zwraca kartę użytkownikowi *Eject Card*.
- **Użytkownik** odbiera kartę *Collect Card*.
- **ATM** wyświetla końcowy komunikat *Thank You*, kończąc interakcję.

5. Zadanie 2 - przykład dla czytelni.



Rysunek 2: Diagram sekwencji systemu czytelni przedstawiający procesy logowania, przeglądania katalogów, wypożyczania książek przez użytkownika i jak to wpływa kolejno na system, oraz proces akceptacji wypożyczenia przed administratora oraz kolejne procesy wykonywane kolejno przez system.

5.1. Logowanie użytkownika

Proces rozpoczyna się, gdy **Użytkownik** otwiera stronę logowania systemu bibliotecznego:

- **Użytkownik** otwiera stronę logowania i wprowadza dane uwierzytelniające, wysyłając je do **Interfejsu Web**.
- **Interfejs Web** przekazuje dane logowania do **Systemu**.
- **System** przesyła dane do **Bazy danych** w celu ich weryfikacji.
- **Baza danych** zwraca wynik autoryzacji do **Systemu**.

Na diagramie przedstawiono alternatywę `wyniku autoryzacji`:

1. **Dane poprawne** - **System** przekierowuje użytkownika do ekranu katalogu.
2. **Dane błędne** - **Interfejs Web** wyświetla komunikat o błędzie logowania.

5.2. Przeglądanie katalogu

Po poprawnym logowaniu użytkownik może rozpocząć przeglądanie dostępnych zasobów systemu czytelniczego:

- **Użytkownik** wybiera katalog, wysyłając odpowiednie żądanie do **Interfejsu Web**.
- **Interfejs Web** przekazuje żądanie do **Systemu**.
- **System** pobiera listę książek z **Bazy danych**.
- **Baza danych** zwraca listę książek, którą **Interfejs Web** prezentuje użytkownikowi.

Następnie użytkownik może zawęzić wyniki:

- **Użytkownik** ustawia filtry lub wpisuje kryteria wyszukiwania.
- **Interfejs Web** przekazuje filtry do **Systemu**.
- **System** pobiera z **Bazy danych** listę książek spełniających podane kryteria.
- Otrzymane dane prezentowane są użytkownikowi jako lista przefiltrowana.

5.3. Wypożyczenie książki

Gdy użytkownik wybierze interesującą go pozycję:

- **Użytkownik** wybiera książkę i inicjuje proces wypożyczenia poprzez akcję *Wypożycz*.
- **Interfejs Web** przekazuje żądanie wypożyczenia do **Systemu**.
- **System** sprawdza dostępność wybranej pozycji w **Bazie danych**.

Następnie pojawia się alternatywa `dostępność książki`:

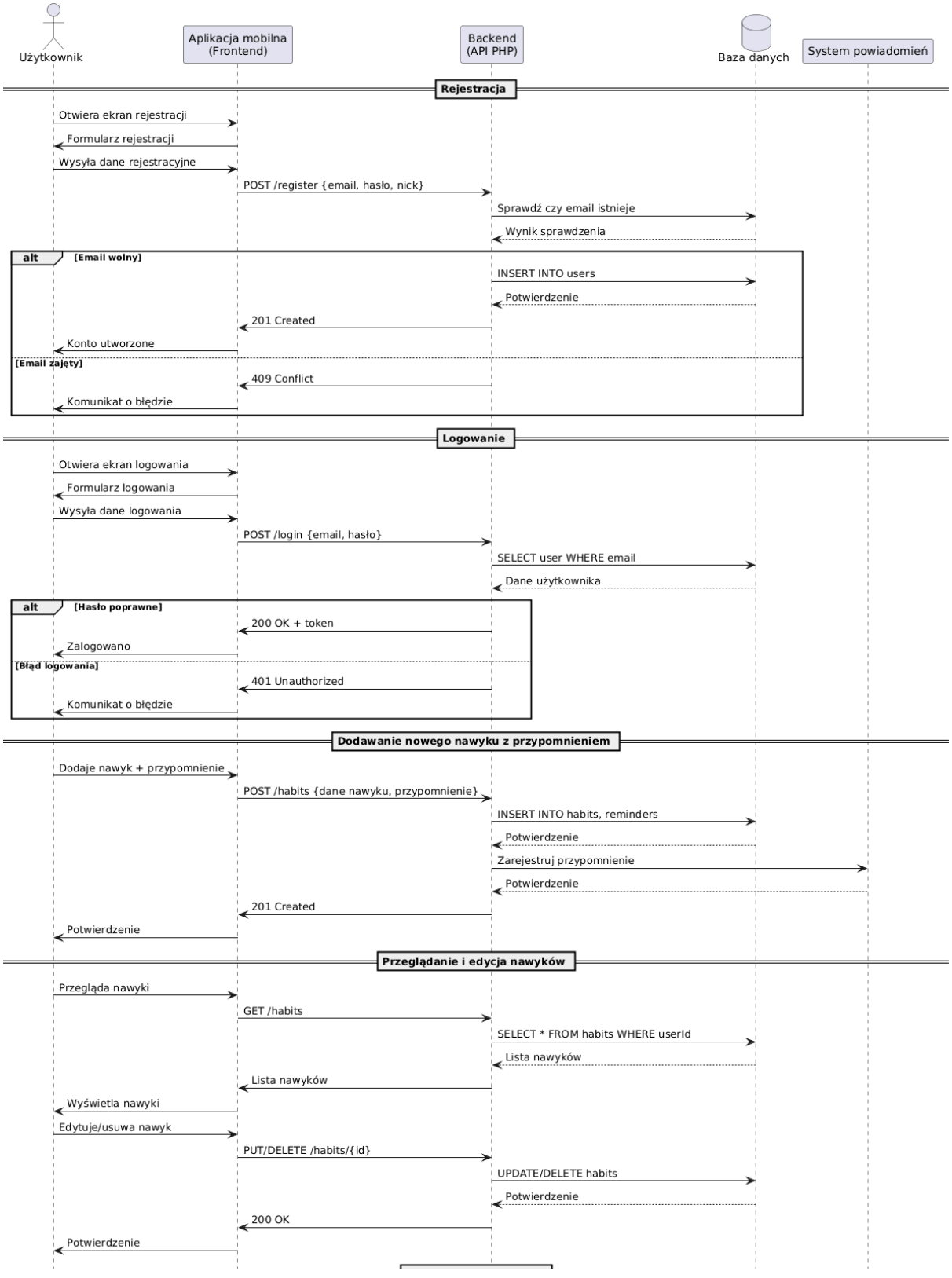
1. **Książka dostępna**:
   - **System** dodaje nowe wypożyczenie ze statusem *oczekujące*.
   - **System** powiadamia **Administratora** o nowym wypożyczeniu.
   - **Interfejs Web** wyświetla komunikat: *Wypożyczenie oczekuje na akceptację*.
2. **Książka niedostępna**:
   - **Interfejs Web** prezentuje użytkownikowi komunikat: *Książka niedostępna*.

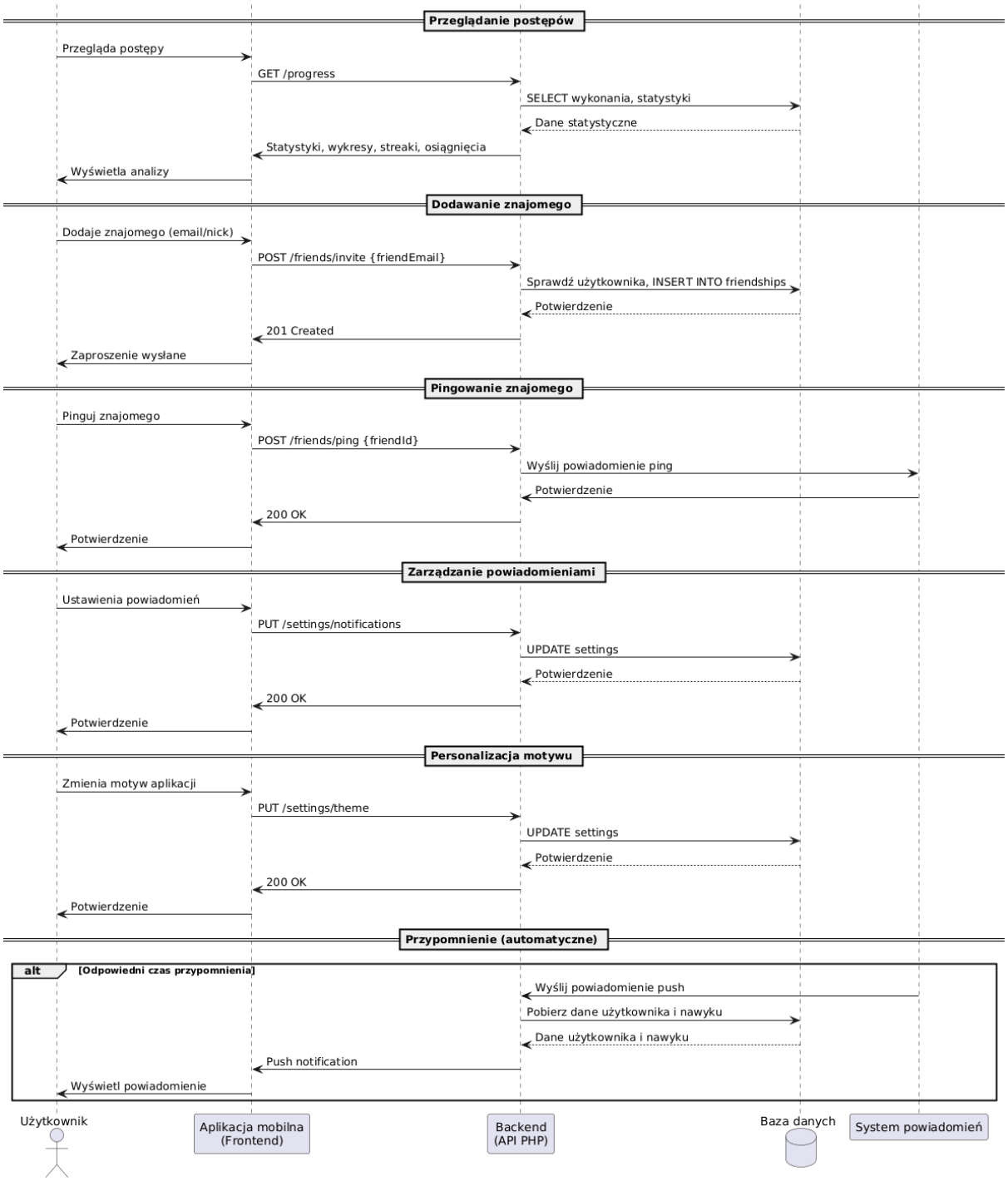
5.4. Akceptacja wypożyczenia

Proces kończy się po stronie administracyjnej:

- **Administrator** akceptuje wypożyczenie w systemie.
- **System** aktualizuje status wypożyczenia na *Wypożyczona* w **Bazie danych**.
- **System** wysyła powiadomienie do **Użytkownika** o zaakceptowaniu wypożyczenia.

6. Zadanie 3 - przykład dla projektu zespołowego.


Użytkownik
Aplikacja mobilna (Frontend)
Backend (API PHP)
Baza danych
System powiadomień
Rejestracja
Otwiera ekran rejestracji
Formularz rejestracji
Wysyła dane rejestracyjne
POST /register {email, hasło, nick}
Sprawdź czy email istnieje
Wynik sprawdzenia
alt
[Email wolny]
INSERT INTO users
Potwierdzenie
201 Created
Konto utworzone
[Email zajęty]
409 Conflict
Komunikat o błędzie
Logowanie
Otwiera ekran logowania
Formularz logowania
Wysyła dane logowania
POST /login {email, hasło}
SELECT user WHERE email
Dane użytkownika
alt
[Hasło poprawne]
200 OK + token
Zalogowano
[Błąd logowania]
401 Unauthorized
Komunikat o błędzie
Dodawanie nowego nawyku z przypomnieniem
Dodaje nawyk + przypomnienie
POST /habits {dane nawyku, przypomnienie}
INSERT INTO habits, reminders
Potwierdzenie
Zarejestruj przypomnienie
Potwierdzenie
201 Created
Potwierdzenie
Przeglądanie i edycja nawyków
Przegląda nawyki
GET /habits
SELECT * FROM habits WHERE userId
Lista nawyków
Lista nawyków
Wyświetla nawyki
Edytuje/usuwa nawyk
PUT/DELETE /habits/{id}
UPDATE/DELETE habits
Potwierdzenie
200 OK
Potwierdzenie



6.1. Rejestracja użytkownika

Proces rozpoczyna się, gdy **Użytkownik** otwiera ekran rejestracji w aplikacji mobilnej:

- **Użytkownik** otwiera formularz rejestracyjny i wypełnia wymagane pola.
- Aplikacja mobilna (**Frontend**) wysyła dane rejestracyjne do **Backendu** poprzez żądanie `POST /register`.

- **Backend** sprawdza w **Bazie danych**, czy podany adres e-mail już istnieje.
- **Baza danych** zwraca wynik weryfikacji.

Występuje tu konstrukcja alternatywna `wyniku sprawdzania`:

1. **Email wolny**:
    - **Backend** tworzy nowy rekord użytkownika w tabeli `users`.
    - **Baza danych** potwierdza poprawne dodanie użytkownika.
    - **Frontend** otrzymuje odpowiedź `201 Created`, informując o utworzonym koncie użytkownika.
2. **Email zajęty**:
    - **Backend** zwraca komunikat błędu `409 Conflict`.
    - **Frontend** informuje użytkownika o duplikacie adresu e-mail.

6.2. Logowanie użytkownika

Aby uzyskać dostęp do aplikacji, użytkownik wykonuje:

- Otwarcie ekranu logowania i wprowadzenie danych.
- **Frontend** wysyła żądanie `POST /login`.
- **Backend** pobiera dane użytkownika z **Bazy danych**.

Następnie występuje alternatywa:

1. **Hasło poprawne**:
    - **Backend** zwraca kod `200 OK` wraz z tokenem sesyjnym.
    - Aplikacja informuje o poprawnym logowaniu.
2. **Hasło błędne**:
    - **Backend** zwraca kod `401 Unauthorized`.
    - **Frontend** wyświetla komunikat o błędzie.

6.3. Dodawanie nowego nawyku z przypomnieniem

Po zalogowaniu użytkownik może stworzyć nowy nawyk:

- **Użytkownik** uzupełnia formularz nowego nawyku i opcjonalne przypomnienie.
- **Frontend** wysyła żądanie `POST /habits`.
- **Backend** dodaje rekordy do tabel `habits` oraz `reminders` w **Bazie danych**.
- **System powiadomień** rejestruje nowe przypomnienie.
- Aplikacja otrzymuje odpowiedź `201 Created`.

6.4. Przeglądanie i edycja nawyków

Użytkownik może przeglądać lub modyfikować swoje nawyki:

- Żądanie `GET /habits` trafia do **Backendu**.
- **Backend** pobiera listę nawyków z **Bazy danych**.
- **Frontend** wyświetla pobraną listę.

W przypadku edycji lub usunięcia:

- **Frontend** wysyła żądanie `PUT/DELETE /habits/{id}`.
- **Backend** dokonuje odpowiedniej operacji w **Bazie danych**.
- Po zakończeniu przetwarzania zwracany jest kod `200 OK`.

6.5. Przeglądanie postępów

Użytkownik może przeanalizować swoje statystyki:

- **Frontend** wykonuje żądanie `GET /progress`.
- **Backend** pobiera dane statystyczne z **Bazy danych**.
- Aplikacja prezentuje użytkownikowi wykresy, streaki i osiągnięcia.

6.6. Dodawanie znajomego

W celu zwiększenia motywacji użytkownik może dodać znajomego:

- **Frontend** wysyła żądanie `POST /friends/invite`.
- **Backend** sprawdza istnienie użytkownika i dodaje relację w tabeli `friendships`.
- **Frontend** otrzymuje odpowiedź `201 Created`.

6.7. Pingowanie znajomego

Użytkownik może wysłać szybkie przypomnienie do znajomego:

- Wysyłane jest żądanie `POST /friends/ping`.
- **Backend** inicjuje powiadomienie w **Systemie powiadomień**.
- **Frontend** otrzymuje odpowiedź `200 OK`.

6.8. Zarządzanie powiadomieniami

Użytkownik może dostosować parametry przypomnień:

- **Frontend** wysyła żądanie `PUT /settings/notifications`.
- **Backend** aktualizuje ustawienia w **Bazie danych**.
- Zwracana jest odpowiedź `200 OK`.

6.9. Personalizacja motywu aplikacji

Użytkownik może zmieniać motyw wizualny:

- Żądanie `PUT /settings/theme` trafia do **Backendu**.
- **Baza danych** aktualizuje preferencje.
- Aplikacja otrzymuje odpowiedź `200 OK`.

6.10. Przypomnienia (automatyczne)

System obsługuje wysyłanie przypomnień nawet bez akcji użytkownika:

- **System powiadomień** cyklicznie sprawdza ustawienia i czas przypomnień.
- W przypadku spełnienia warunków wysyłane jest powiadomienie push.
- **Backend** pobiera dane użytkownika i nawyku celem generacji treści powiadomienia.
- **Frontend** prezentuje powiadomienie użytkownikowi.

7. Diagram interakcji sekwencji i współpracy - omówienie różnic

Diagramy interakcji w notacji UML można podzielić na dwa główne typy: diagram sekwencji oraz diagram współpracy (zwany również diagramem komunikacji). Oba typy modelują sposób, w jaki obiekty wzajemnie oddziałują, jednak każdy z nich eksponuje inne aspekty analizowanego procesu. W niniejszej sekcji omówiono kluczowe cechy obu diagramów oraz wskazano fundamentalne różnice między nimi.

7.1. Charakterystyka diagramu sekwencji

- Koncentruje się na przebiegu zdarzeń w czasie, ukazując chronologiczną kolejność komunikatów.
- Wykorzystuje pionowe linie życia, które wizualizują czas istnienia obiektów w trakcie interakcji.
- Komunikaty ułożone są liniowo od góry do dołu, co pozwala prześledzić pełen przebieg procesu.
- Podkreśla rytm, czas trwania i zależności czasowe między operacjami.
- Jest szczególnie przydatny w analizie procesów logicznych i dynamicznych przepływów danych.

7.2. Charakterystyka diagramu współpracy

- Skupia się na strukturze komunikacji między obiektami, a nie na czasie wykonania.
- Komunikaty opisane są numeracją odzwierciedlającą kolejność operacji, ale nie ich czasową liniowość.
- Umożliwia analizę organizacji i sieci powiązań między obiektami.
- Przedstawia obiekty w układzie przestrzennym, co pozwala ocenić ich relacje oraz stopień zależności.
- Często stosowany podczas projektowania architektury systemu lub sieci współpracujących agentów.

7.3. Porównanie obu typów diagramów

- **Perspektywa czasu:** Diagram sekwencji akcentuje oś czasu, natomiast diagram współpracy przedstawia relacje strukturalne, bez dokładnego odwzorowania przebiegu czasowego.

- **Priorytet wizualizacji:** Diagram sekwencji eksponuje kolejność wywołań, a diagram współpracy - zależności między obiektami.
- **Złożoność interpretacji:** Diagram sekwencji jest łatwiejszy do odczytania w przypadku długich procesów. Diagram współpracy lepiej prezentuje rozbudowane relacje między obiektami.
- **Różne zastosowania projektowe:** Diagram sekwencji zalecany jest dla analizy procesów użytkownika (use-case), a diagram współpracy dla projektowania architektury i struktury systemu.

7.4. Kiedy stosować każdy z diagramów

- Diagram sekwencji:
  - analizowanie przepływów czasowych i zależności operacji,
  - wykrywanie błędów w logicznym następstwie czynności,
  - modelowanie procesów użytkownika w systemach interaktywnych,
  - analiza zachowania systemów czasu rzeczywistego.
- Diagram współpracy:
  - analizowanie zależności obiektów w architekturze systemu,
  - projektowanie komunikujących się modułów,
  - optymalizacja struktury interakcji,
  - modelowanie systemów wieloagentowych.

8. Wnioski

Diagramy sekwencji i współpracy stanowią kluczowe narzędzia w inżynierii oprogramowania, umożliwiając zrozumienie dynamiki działania systemu oraz relacji między jego elementami. Ich stosowanie przynosi liczne korzyści na etapie projektowania, dokumentowania i analizy procesów informatycznych.

8.1. Zastosowania diagramów interakcji

- Dokumentacja zachowania systemów w czasie rzeczywistym.
- Analiza funkcjonalności z poziomu użytkownika (modelowanie przypadków użycia).
- Projektowanie interakcji między modułami i podsystemami.
- Wspieranie analizy wymagań oraz walidacja kompletności specyfikacji.

8.2. Celowość ich wykorzystania w procesie informatycznym

- Pozwalają wcześnie wykryć nieścisłości w projektowaniu logiki systemu.
- Stanowią podstawę do tworzenia kodu źródłowego i architektury systemowej.
- Zapewniają formalny sposób komunikacji pomiędzy członkami zespołu projektowego.
- Ułatwiają analizę i optymalizację przepływu danych i komunikatów.

8.3. Znaczenie dla bezpieczeństwa i niezawodności systemu

- Umożliwiają identyfikację potencjalnych punktów awarii lub przeciążenia.
- Wspierają analizę ryzyka poprzez modelowanie scenariuszy wyjątkowych.
- Pozwalają przewidzieć konsekwencje błędnych interakcji między komponentami.

8.4. Rola w procesie implementacji i testowania

- Ułatwiają projektowanie struktur klas i interfejsów API.
- Tworzą podstawę dla testów jednostkowych i integracyjnych.
- Pozwalają zasymulować różne ścieżki wykonywania programu.
- Wspomagają automatyzację testów poprzez mapowanie scenariuszy.

8.5. Podsumowanie wniosków

- Diagramy interakcji są niezbędne w procesie projektowania dynamicznego zachowania systemu.
- Zapewniają pełną przejrzystość procesów oraz hierarchii komunikatów.
- Poprawiają efektywność projektowania, implementacji i weryfikacji oprogramowania.
- Ich stosowanie przekłada się na wyższą jakość, stabilność i zrozumiałość systemu informatycznego.