

# Dokumentacja projektu pt. „Aplikacja wspomagająca inwestycje za pomocą funduszy inwestycyjnych”

Realizowanego w ramach ćwiczeń z przedmiotu Inżynieria  
Oprogramowania, semestr zimowy 2022/23 AGH



*Logotyp projektu*

**Autorzy:**

Kacper Papuga, Jakub Nowak, Wiktor Paleczny

# Spis treści

1.	Specyfikacja systemu .....	3
2.	Opis wybranej metodyki wytwarzania.....	6
3.	Słownik pojęć i terminów.....	7
4.	Diagram przypadków użycia.....	8
4.	Wymagania funkcjonalne i niefunkcjonalne oraz zarys wymagań biznesowych.....	19
5.	Diagram wymagań.....	22
6.	Wykaz zastosowanych framework'ów, bibliotek, środowisk oraz innych technologii i rozwiązań informatycznych .....	23
7.	Diagramy aktywności.....	26
8.	Diagram klas.....	35
9.	Diagram sekwencji.....	37
10.	Diagram komponentów .....	45
11.	Diagram wdrożenia.....	46
12.	Przykłady współpracy zespołu .....	47
13.	Testy jednostkowe.....	50
14.	Przykłady refaktoryzacji kodu.....	53
15.	Przykład wzorca projektowego.....	56
16.	Instrukcja użytkowania programu .....	58
17.	Podsumowanie .....	63

## 1. Specyfikacja systemu

### 1.1 Koncepcja:

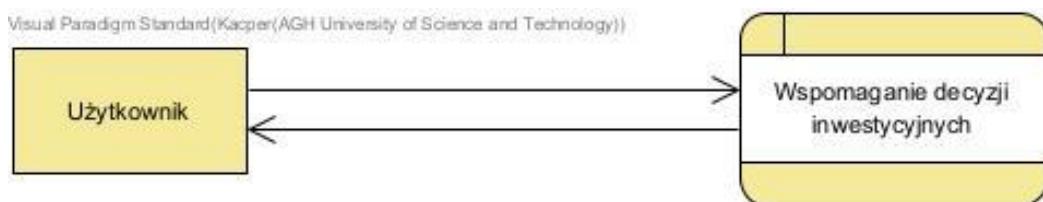
Celem projektowym jest stworzenie aplikacji webowej, która umożliwia wspomaganie decyzji inwestycyjnych poprzez zbieranie i prezentowanie danych giełdowych udostępnianych przez inne serwery. Dane odzwierciedlają tym samym rzeczywistą sytuację rynkową, ponieważ pochodzą z realnych źródeł. Główną koncepcją podczas projektowania oraz implementowania systemu było skupienie się na dwóch elementach: części backend'owej aplikacji, której zadaniem jest komunikacja z zewnętrznymi serwerami w celu pozyskania danych oraz komunikację z bazą danych w celu ich przechowywania, a także komunikacja z częścią frontend'ową projektu odpowiedzialną za prezentowanie danych.

### 1.2 Podział systemu:

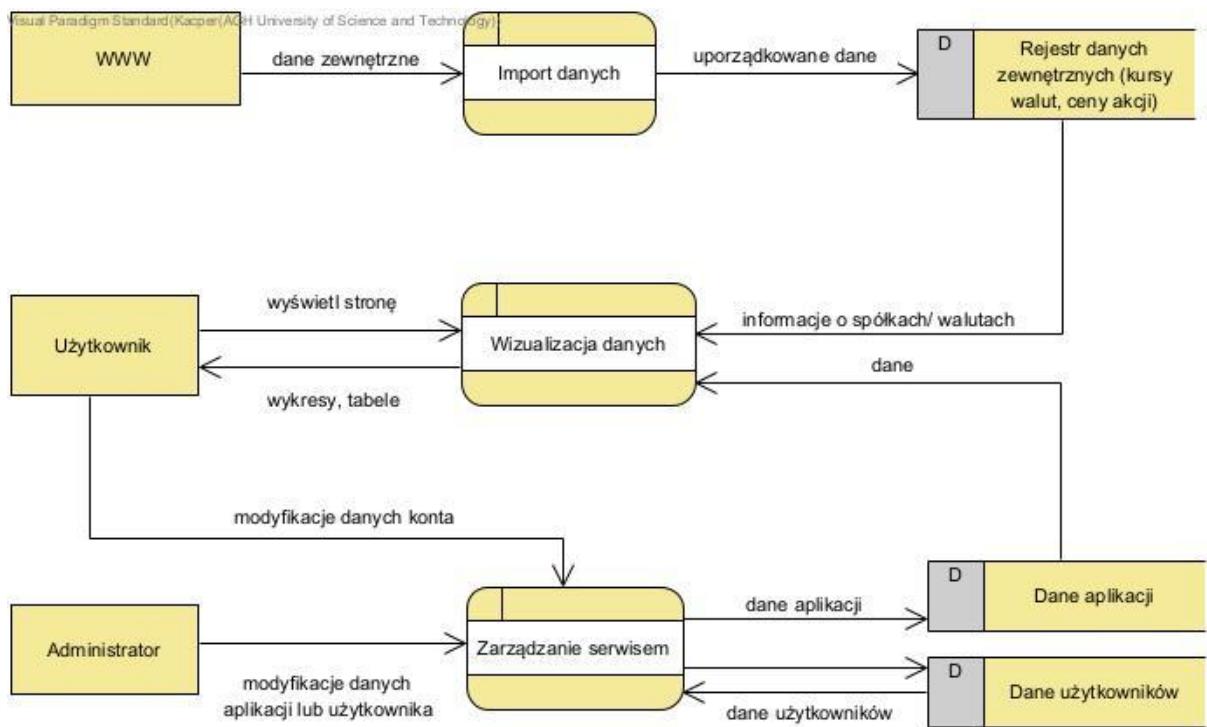
- Serwer aplikacji (backend)** – mechanizm pobierania danych z zewnętrznych serwerów, komunikacja z bazą danych w celu zapisu, modyfikacji oraz pozyskiwania danych, obsługa mechanizmów aplikacji dotyczących autoryzacji
- Aplikacja kliencka (frontend)** – mechanizm odpowiedzialny za komunikację aplikacja-serwer w celu pozyskiwania danych oraz aplikacja-użytkownik w celu udostępniania danych oraz stworzenia interfejsu użytkownika
- Baza danych** – umożliwia przechowywanie danych potrzebnych do poprawnego działania aplikacji (np. dane użytkowników) oraz

### 1.3 Przepływ danych:

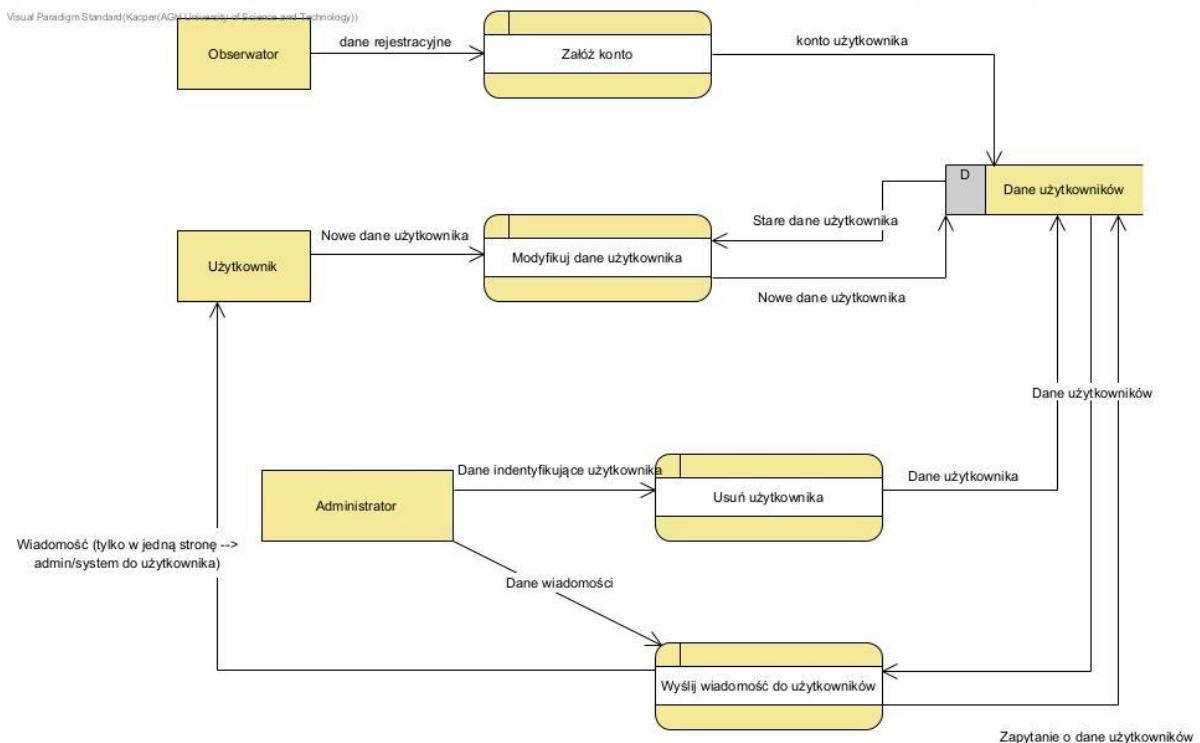
Najogólniejszy diagram przepływu danych



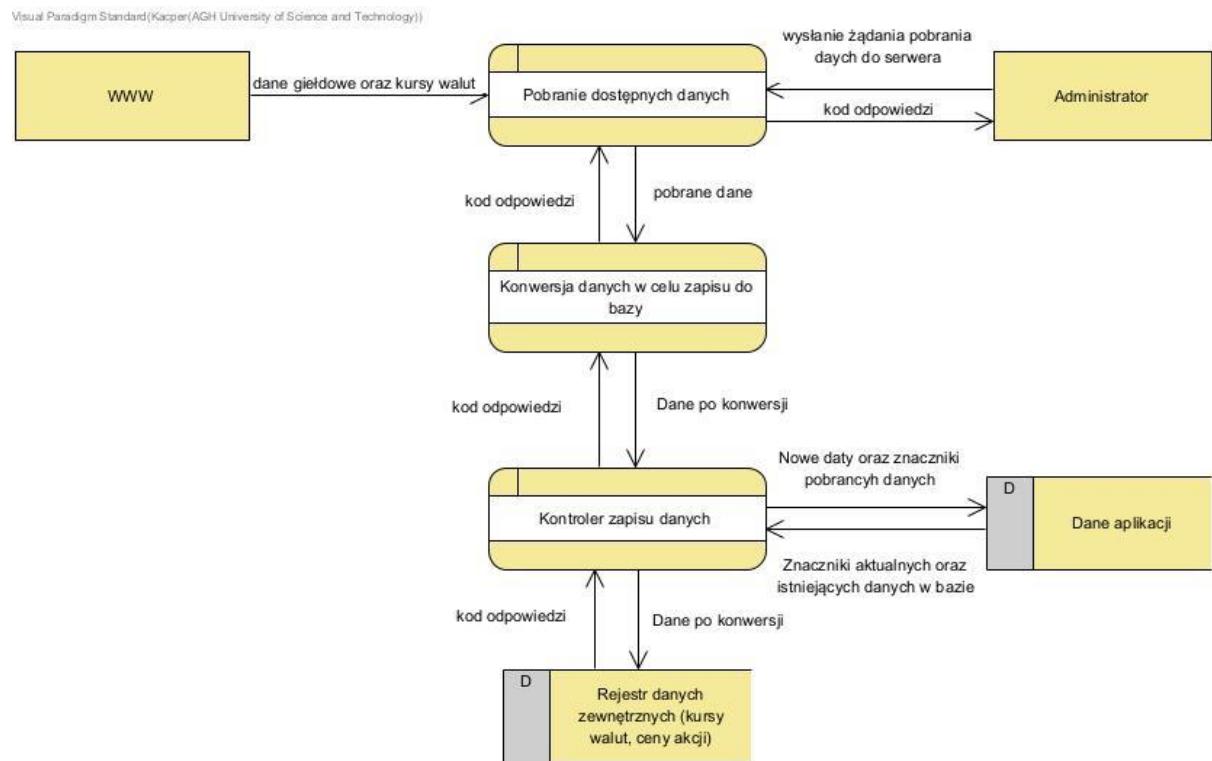
### Dekompozycja diagramu – poziom 1



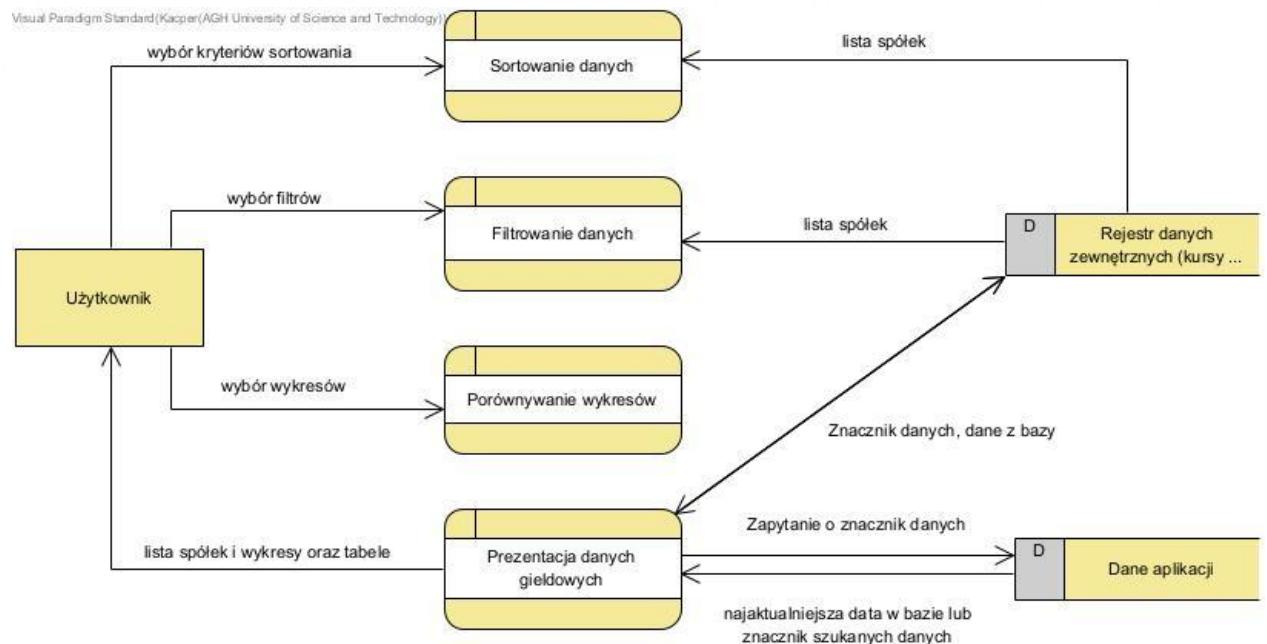
### Dekompozycja diagramu przepływu danych przy zarządzaniu serwisem – poziom 2



## Dekompozycja diagramu przepływu danych przy importowaniu danych – poziom 2



## Dekompozycja diagramu przepływu danych przy wizualizacji danych – poziom 2

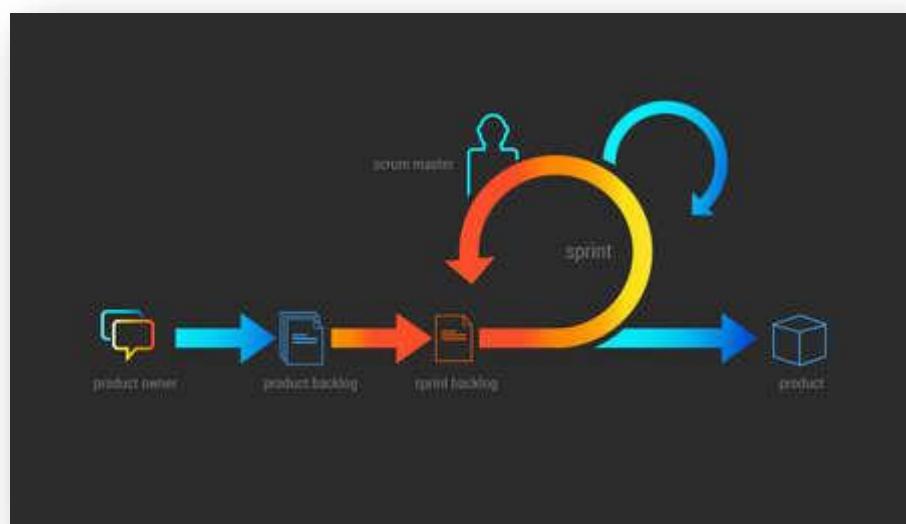


## 2. Opis wybranej metodyki wytwarzania

Manifest AGILE opisujący metodologię zwinnego wytwarzania oprogramowania ramach której funkcjonuje wybrana przez nas metodologia SCRUM opiera się na czterech wartościach opisujących co należy najbardziej cenić w praktyce zawodowej podczas całokształtu procesu wytwarzania oprogramowania:

1. **Ludzie i interakcje** od procesów i narzędzi
2. **Działające oprogramowanie** od szczegółowej dokumentacji
3. **Współpracę z klientem** od negocjacji umów
4. **Reagowanie na zmiany** od realizacji założonego planu

Zwinne wytwarzanie oprogramowania sprawdza się w szczególności w środowisku, w którym występują częste i nieprzewidywalne zmiany, co ze względu na niewielkie doświadczenie członków zespołu w zakresie programowania stron internetowych może być powszechnym zjawiskiem.



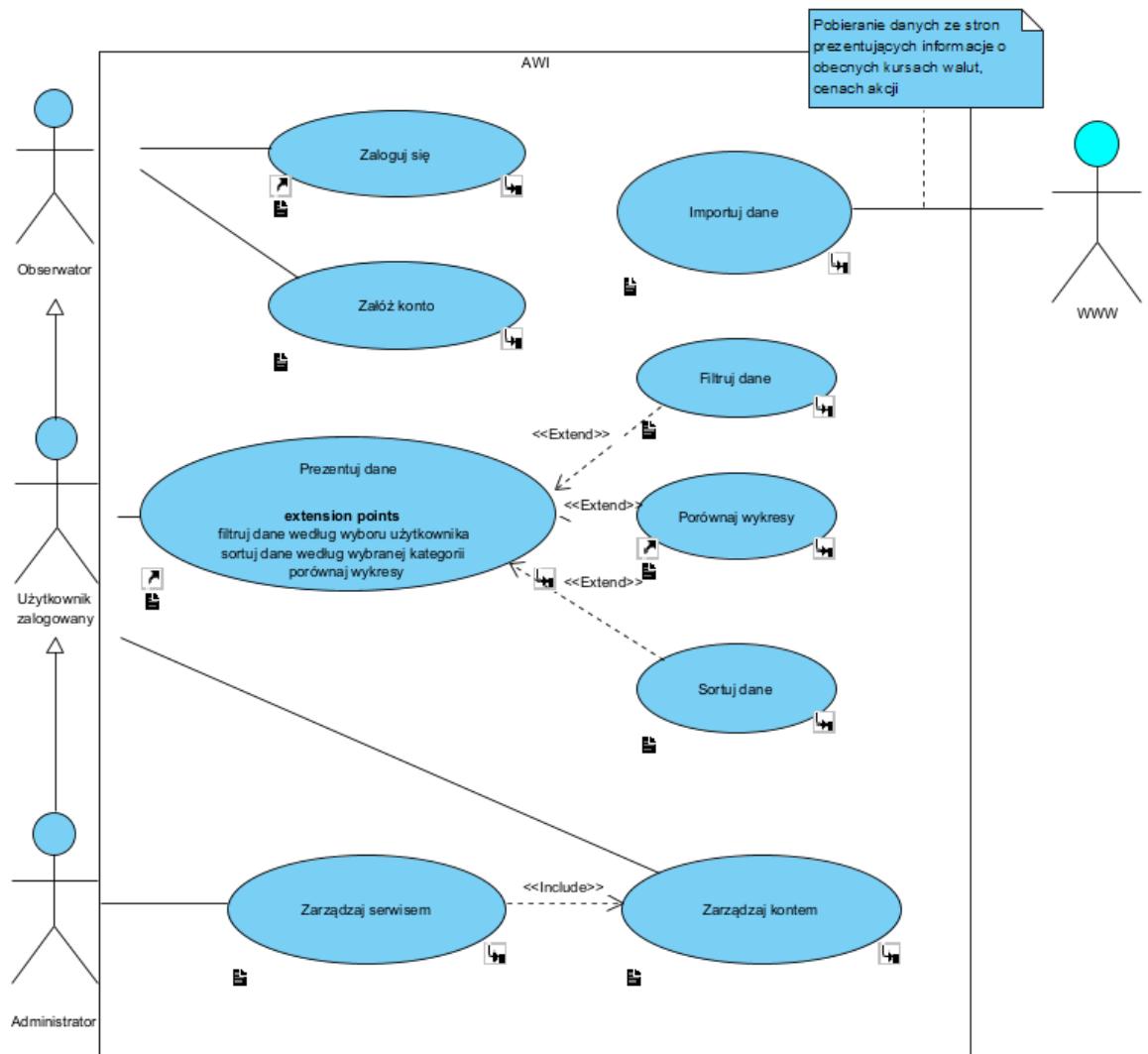
Metoda SCRUM, ma właśnie na celu tworzenie rozwiązań w takim, podatnym na zmiany, środowisku. Zorientowana jest również na interakcje i relacje, empiryzm, czyli podejmowanie decyzji na podstawie nabyciego doświadczenia i obserwacji oraz przede wszystkim na **przyrostowym, ewolucyjnym podejściu do wytwarzania oprogramowania**. Jako trzyosobowy zespół, uznaliśmy, że taka metodologia nakierowana na komunikację, zmienność oraz krótkie cykle czasowe programowania, kończące się podsumowaniem naszej pracy i opracowaniem planu na dalszy rozwój będzie najlepszym modelem pracy.

### **3. Słownik pojęć i terminów**

**Pojęcia dotyczące prezentowanych na stronie danych:**

- **EffectiveDate** – data publikacji
- **Currency** – nazwa waluty
- **Code** – kod waluty, skrót
- **Mid** – przeliczony kurs średni waluty (dotyczy)
- **Change** - zmiana procentowej wartości indeksu
- **MinimalRate** - najniższa cena dnia
- **MinimalRate** -najwyższa cena dnia

#### 4. Diagram przypadków użycia



#### 1. Zaloguj się

ID: UC13

**PU "Zaloguj się" jest implementacją funkcjonalności weryfikacji użytkownika w systemie.**  
Wymaga posiadania konta oraz podania prawidłowych danych logowania w celu weryfikacji.  
**Użytkownicy zalogowani posiadają dostęp do zwiększonej funkcjonalności systemu.**

<b>Justification</b>	Konieczność autoryzacji użytkowników, w celu udostępniania wybranych zasobów tylko zalogowanym użytkownikom
<b>Primary Actors</b>	Obserwator
<b>Level</b>	User
<b>Complexity</b>	Medium
<b>Use Case Status</b>	Complete

<b>Implementation Status</b>	Completed
<b>Preconditions</b>	Użytkownik znajduje się w oknie startowym aplikacji oraz wybiera opcję "Zaloguj się"
<b>Post-conditions</b>	Po podaniu poprawnych danych, użytkownik zostaje zalogowany w systemie i przeniesiony do okna wyboru czynności w aplikacji. Dostaje dostęp do zwiększonej liczby funkcjonalności aplikacji
<b>Author</b>	Kacper Papuga
<b>Assumptions</b>	Użytkownik posiada konto, które zostało aktywowane poprzez link aktywacyjny (potwierdzenie adresu e-mail)

## 1.1 Scenariusze

### 1.1.1 Scenariusz główny

1. Użytkownik wybiera opcję "Zaloguj się" z okna startowego
2. System przenosi użytkownika do okna logowania
3. Użytkownik podaje dane
4. System weryfikuje podane dane
5. Weryfikacja przebiega poprawnie i system przenosi użytkownika do okna wyboru dalszych czynności

### 1.1.2.Scenariusz wyjątku

1. W punkcie 5 scenariusza głównego użytkownik nie zostaje zweryfikowany pomyślnie
2. System wyświetla odpowiedni komunikat błędu
3. System umożliwia dalszą próbę logowania

2. Załóż konto

ID: UC01

**PU umożliwia obserwatorowi zarejestrowanie się w aplikacji.** Dzięki założeniu konta, system będzie mógł zweryfikować użytkownika oraz udostępnić mu zwiększoną funkcjonalność. PU kończy się **w momencie aktywacji konta** (potwierdzenia adresu e-mail). **Użytkownik dostaje dostęp do takich funkcjonalności systemu jak: sprawdzenie oraz filtrowanie kursów walut, cen akcji giełdowych, czy zmiany swoich danych.**

<b>Justification</b>	Konieczność umożliwienia obserwatorom założenia konta w celu udostępnienia funkcjonalności aplikacji
<b>Primary Actors</b>	 Obserwator
<b>Level</b>	User
<b>Complexity</b>	Medium

<b>Use Case Status</b>	Complete
<b>Implementation Status</b>	Scheduled
<b>Preconditions</b>	Obserwator znajduje się na stronie głównej aplikacji oraz wybiera opcję "Zarejestruj się".
<b>Post-conditions</b>	Konto o podanych danych zostaje zapisane w bazie danych, ale konto pozostaje nieaktywne (nie można się na nie zalogować) do momentu potwierdzenia go poprzez kliknięcie w link wysłany na wskazany adres e-mail
<b>Author</b>	Kacper Papuga
<b>Assumptions</b>	Obserwator posiada konto na poczcie elektronicznej oraz akceptuje regulamin aplikacji.

## 2.1 Scenariusze

### 2.1.1. Scenariusz główny

1. Obserwator wybiera opcję z okna głównego aplikacji "Zarejestruj się"
2. System przenosi użytkownika do okna rejestracji.
3. Użytkownik podaje dane rejestracji
4. Następnie wybiera opcję "Załóż konto"
5. System zapisuje użytkownika w bazie danych jako nieaktywne konto - bez możliwości zalogowania
6. Użytkownik zostaje przeniesiony do okna logowania

### 2.1.2. Scenariusz wyjątku

1. W kroku 3 scenariusza głównego użytkownik podaje login/adres e-mail zajęty przez innego użytkownika
2. System wyświetla komunikat o zajętym loginie/adresie e-mail
3. Powrót do kroku 3 scenariusza głównego

## 3. Importuj dane

ID: UC06

**PU umożliwia pobieranie danych z innych serwisów internetowych** i zapisywanie ich do bazy danych, z której nasepnie pobierane są w celu ich wyświetlania na stronie. PU odpowiedzialny jest za pobieranie takich danych jak: kursy walut oraz ceny akcji. Dane te umożliwiają następnie m.in prezentację danych w postaci graficznej, tabelarycznej, czy dokonywanie na nich obliczeń.

<b>Justification</b>	Konieczność pobierania rzeczywistych danych giełdowych
<b>Primary Actors</b>	WWW
<b>Level</b>	Summary
<b>Complexity</b>	High
<b>Use Case Status</b>	Complete
<b>Implementation Status</b>	Scheduled
<b>Preconditions</b>	Znalezienie rzeczywistych danych możliwych do zimportowania do bazy danych
<b>Post-conditions</b>	Dane dostępne dla aplikacji poprzez komunikację z bazą danych w celu wykorzystania ich do innych funkcjonalności systemu
<b>Author</b>	Kacper Papuga
<b>Assumptions</b>	Wybór odpowiedniej do projektu bazy danych oraz połączenie z nią serwera aplikacji, istnienie rzeczywistych danych możliwych do zimportowania do bazy danych

### 3.1 Scenariusze

#### 3.1.1 Scenariusz główny

1. Administrator lub system inicjuje pobieranie danych z innych (wcześniej wskazanych serwisów)
2. System zapisuje pobrane dane w bazie danych
3. System aktualizuje dane wyświetlane w aplikacji zgodnie z zawartością bazy danych

#### 3.1.2 Scenariusz wyjątku

1. W kroku 1 scenariusza głównego system nie inicjuje pobierania danych
2. System informuje o błędzie, który uniemożliwił pobranie danych

#### 3.1.3 Scenariusz wyjątku

1. W kroku 2 scenariusza głównego system nie zapisuje odpowiednio danych
2. System informuje o błędzie, który uniemożliwił poprawny zapis danych

#### 3.1.4 Scenariusz wyjątku

1. W kroku 3 scenariusza głównego system nie aktualizuje danych w aplikacji
2. System informuje o błędzie, który uniemożliwił poprawne wyświetlenie zaktualizowanych danych w aplikacji

## 4. Prezentuj dane

ID: UC02

PU umożliwia dostęp użytkownikom zalogowanym do danych dostępnych na stronie aplikacji w postaci prezentacji tabelarycznej i graficznie danych dotyczących cen akcji spółek notowanych na giełdzie oraz kursów walut. Przypadek ten jest rozszerzany przez inne przypadki użycia służących do porządkowania wyświetlanych danych

<b>Justification</b>	Konieczność prezentowania danych znajdujących się w bazie danych
<b>Primary Actors</b>	👤 Użytkownik zalogowany
<b>Level</b>	User
<b>Complexity</b>	Low
<b>Use Case Status</b>	Base
<b>Implementation Status</b>	Completed
<b>Preconditions</b>	Znajdowanie się na odpowiedniej podstronie aplikacji lub wybór z podstrony wyboru lub paska nawigacji odpowiedniej podstrony
<b>Post-conditions</b>	Wyświetlenie wybranych danych (spółek/walut) z bazy danych
<b>Author</b>	Jakub Nowak
<b>Assumptions</b>	Użytkownik jest zalogowany w aplikacji, poprawne dane znajdują się w bazie danych

### 4.1. Scenariusze

#### 4.1.1. Scenariusz główny

1. Użytkownik wybiera z menu jedną z opcji dotyczących wyświetlenia cen akcji, kursów walut lub porównania wykresów
2. System przekierowuje użytkownika do odpowiedniej podstrony
3. System wyświetla odpowiednie dane

#### 4.1.2. Scenariusz wyjątku

1. System nie wyświetla danych z powodu problemu z ich zimportowaniem
2. System wyświetla komunikat błędu

## 5. Filtruj dane

ID: UC03

Rozszerzenie PU "Prezentuj dane", które umożliwia filtrowanie danych dotyczących cen akcji i/lub kursów walut poprzez podanie informacji o szukanych danych. Informacje te podaje w

postaci daty szukanych danych lub w przypadku cen akcji: nazwy spółki, natomiast w przypadku kursów walut: nazwy waluty.

<b>Justification</b>	Konieczność filtrowania danych w celu zwiększenia wyszukiwania interesujących użytkownika informacji
<b>Primary Actors</b>	👤 Użytkownik zalogowany
<b>Level</b>	User
<b>Complexity</b>	Medium
<b>Use Case Status</b>	Base
<b>Implementation Status</b>	Completed
<b>Preconditions</b>	Użytkownik znajduje się w oknie prezentacji cen akcji lub kursów walut
<b>Post-conditions</b>	Po ustawieniu interesujących użytkownika kryteriów na ekranie pojawią się dane spełniające warunki wyszukiwania
<b>Author</b>	Jakub Nowak
<b>Assumptions</b>	Użytkownik jest zalogowany. Odpowiednie dane znajdują się w bazie danych

## 5.1. Scenariusze

### 5.1.1. Scenariusz główny

1. Użytkownik podaje warunki wyszukiwania
2. System pobiera z bazy danych poszukiwane dane
3. System prezentuje pobrane dane

### 5.1.2. Scenariusz wyjątku

1. Użytkownik podaje kryteria nieobsługiwane przez system
2. System wyświetla komunikat błędu

### 5.1.3. Scenariusz wyjątku

1. Użytkownik próbuje wyszukać nieistniejące w bazie danych dane
2. System zwraca odpowiedni komunikat błędu

## 6. Porównaj wykresy

ID: UC05

PU umożliwia wyświetlenie jednocześnie wykresów akcji dwóch wybranych spółek w taki sposób aby można je było porównać i przeanalizować. Oprócz wykresów akcji możliwe jest również porównanie kursów walut. Możliwe jest wyświetlenie wykresów dla 7 lub 30 dni.

<b>Justification</b>	Konieczność analizy zgromadzonych danych
<b>Primary Actors</b>	👤 Użytkownik zalogowany
<b>Level</b>	User
<b>Complexity</b>	Medium
<b>Use Case Status</b>	Base
<b>Implementation Status</b>	Scheduled
<b>Preconditions</b>	Użytkownik znajduje się na podstronie wykresów oraz wybiera dane spośród możliwych opcji
<b>Post-conditions</b>	Na ekranie wyświetcone zostaje zestawienie wykresów dwóch wybranych spółek lub walut
<b>Author</b>	Wiktor Paleczny
<b>Assumptions</b>	Użytkownik jest zalogowany

### 6.1. Scenariusze

#### 6.1.1. Scenariusz główny

1. Użytkownik wybiera odpowiednie instrumenty które chce porównać
2. System importuje potrzebne dane
3. System generuje wykres

#### 6.1.2. Scenariusz wyjątku

1. System nie generuje wykresu
2. Pojawia się komunikat o braku danych

## 7. Sortuj dane

ID: UC04

PU sortuje listę akcji spółek według następujących kryteriów:

- 1) Nazwy (alfabetycznie)
- 2) Zmiany procentowej wartości indeksu
- 3) Najniższej ceny dnia

4) Najwyższej ceny dnia

Sortowanie kursów walut możliwe jest według:

- 1) Nazwy lub skrótu waluty
- 2) Cen walut

<b>Justification</b>	Konieczność ułatwienia odszukiwania interesujących użytkownika danych spośród wyświetlonych
<b>Primary Actors</b>	 Użytkownik zalogowany
<b>Level</b>	User
<b>Complexity</b>	Medium
<b>Use Case Status</b>	Base
<b>Implementation Status</b>	Completed
<b>Preconditions</b>	Użytkownik znajduje się w panelu wyboru lub wybiera z paska nawigacji odpowiednią podstronę
<b>Post-conditions</b>	Po wybraniu przez użytkownika konkretnej opcji na ekranie wyświetli się posortowana lista
<b>Author</b>	Jakub Nowak
<b>Assumptions</b>	Użytkownik jest zalogowany. Odpowiednie dane znajdują się w bazie danych

## 7.1. Scenariusze

### 7.1.1. Scenariusz główny

1. Użytkownik wybiera kryterium sortowania
2. System pokazuje listę posortowaną według wybranego kryterium

### 7.1.2. Scenariusz wyjątku

1. Użytkownik zaznacza więcej niż jedno kryterium wyboru
2. System soruje względem ostatniego wybranego kryterium

## 8. Zarządzaj serwisem

ID: UC11

Przypadek użycia "Zarządzaj serwisem", dostępny jest **wyłącznie dla użytkownika zalogowanego na koncie administratora**. Umożliwia podejmowanie akcji związanych z serwisem takich jak:

- 1) importowanie danych do bazy danych z innych serwisów
- 2) usunięcie dowolnego konta utworzonego w aplikacji
- 3) wysyłanie wiadomości do wszystkich/ poszczególnych użytkowników

PU włącza przypadek "Zarządzaj kontem", który jest dostępny również dla zalogowanego użytkownika.

<b>Justification</b>	Konieczność możliwości zarządzania aplikacją przez administratora
<b>Primary Actors</b>	Administrator
<b>Level</b>	Summary
<b>Complexity</b>	Medium
<b>Use Case Status</b>	Complete
<b>Implementation Status</b>	Scheduled
<b>Preconditions</b>	Użytkownik zna specyfikę systemu oraz wie jak wykonać poszczególne działania na serwerze
<b>Post-conditions</b>	System wykonał wskazaną akcję oraz zapisał efekt jej działania do bazy danych.
<b>Author</b>	Kacper Papuga
<b>Assumptions</b>	Użytkownik ma dostęp do konta administratora

## 8.1. Scenariusze

### 8.1.1. Scenariusz główny

1. Administrator wykonuje na serwerze akcję importowania danych do bazy danych
2. System wykonuje zadanie aktualizując odpowiednio dane
3. System zwraca odpowiedź o statusie wykonanego zadania

### 8.1.2. Scenariusz alternatywny

1. Administrator wykonuje na serwerze akcję związaną z usunięciem dowolnego konta użytkownika bądź jego modyfikację
2. System weryfikuje poprawność wprowadzenia nowych danych (np. w przypadku zmiany adresu email lub loginu użytkownika)
3. System modyfikuje dane zapisane w bazie danych
4. System zwraca odpowiedź o statusie wykonanego zadania

### 8.1.3. Scenariusz alternatywny 2

1. Administrator wykonuje na serwerze akcję związaną z wysłaniem wiadomości do użytkownika/ów
2. System pobiera adres/y email z bazy danych
3. System wysyła wiadomość do wskazanych użytkowników
4. System zwraca odpowiedź o statusie wykonanego zadania

#### **8.1.4. Scenariusz wyjątku**

1. Administrator wykonuje jedną z powyższych akcji
2. System zwraca kod błędu oraz komunikat o błędzie

### **9. Zarządzaj kontem**

ID: UC10

**PU "Zarządzaj kontem" dostępny jest dla użytkownika zalogowanego.** Umożliwia modyfikację danych podanych podczas rejestracji. Możliwe operacje do wykonania:

- 1) Zmiana hasła**
- 2) Zmiana adres e-mail, bez konieczności ponownego potwierdzenia**
- 3) Usunięcie konta**

<b>Justification</b>	Konieczność umożliwienia modyfikacji danych użytkownika
<b>Primary Actors</b>	👤 Użytkownik zalogowany
<b>Supporting Actors</b>	👤 Administrator
<b>Level</b>	User
<b>Complexity</b>	Medium
<b>Use Case Status</b>	Complete
<b>Implementation Status</b>	Scheduled
<b>Preconditions</b>	Użytkownik musi być zalogowany i znajdować się na podstronie ustawień konta, oraz wybrać jedną z akcji
<b>Post-conditions</b>	Użytkownik zmienił swoje dane w systemie lub został mu wyświetlony komunikat o braku możliwości zmiany (brak zmian w systemie).
<b>Author</b>	Kacper Papuga
<b>Assumptions</b>	Użytkownik posiada konto w aplikacji.

### **9.1. Scenariusze**

#### **9.1.1. Scenariusz główny**

1. Użytkownik wybiera jedną z dostępnych opcji ( Zmień email, Zmień hasło ...)
2. Użytkownik podaje nowe dane (email/ hasło)
3. Następnie potwierdza za pomocą przycisku chęć zmiany danych
4. System zapisuje nowe dane dotyczące zalogowanego użytkownika

### **9.1.2. Scenariusz wyjątku**

1. Użytkownik podaje email, który jest zajęty przez innego użytkownika
2. System nie zapisuje danych oraz informuje o błędzie

### **9.1.3. Scenariusz wyjątku 2**

1. Użytkownik podaje hasło, który jest zbyt krótkie
2. System nie zapisuje danych oraz informuje o błędzie

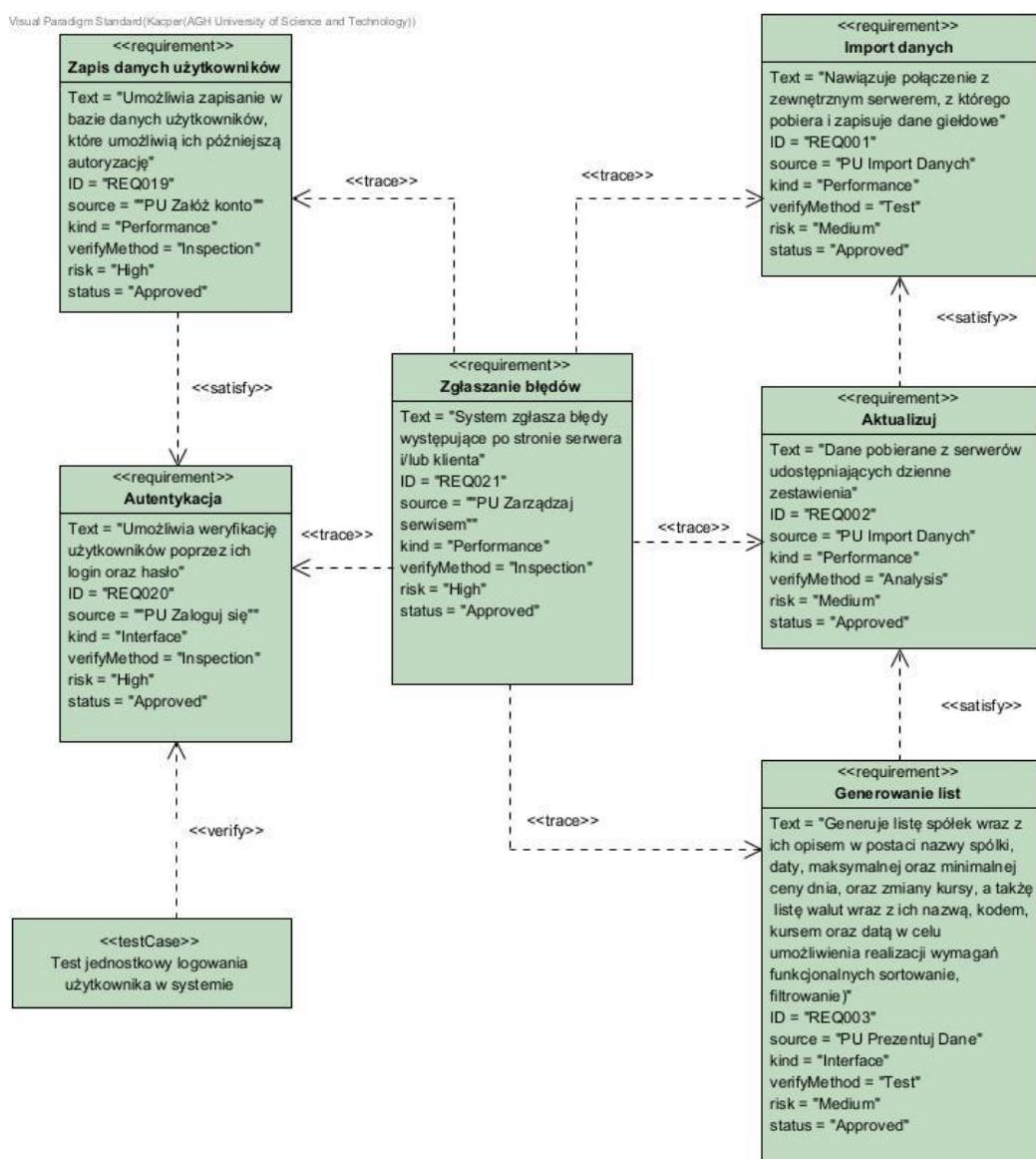
### **9.1.4. Scenariusz wyjątku 3**

1. Po zatwierdzeniu nowych danych system nie zapisuje ich z innych powodów niż zajęty e-mail lub zbyt krótkie hasło
2. System wyświetla komunikat błędu

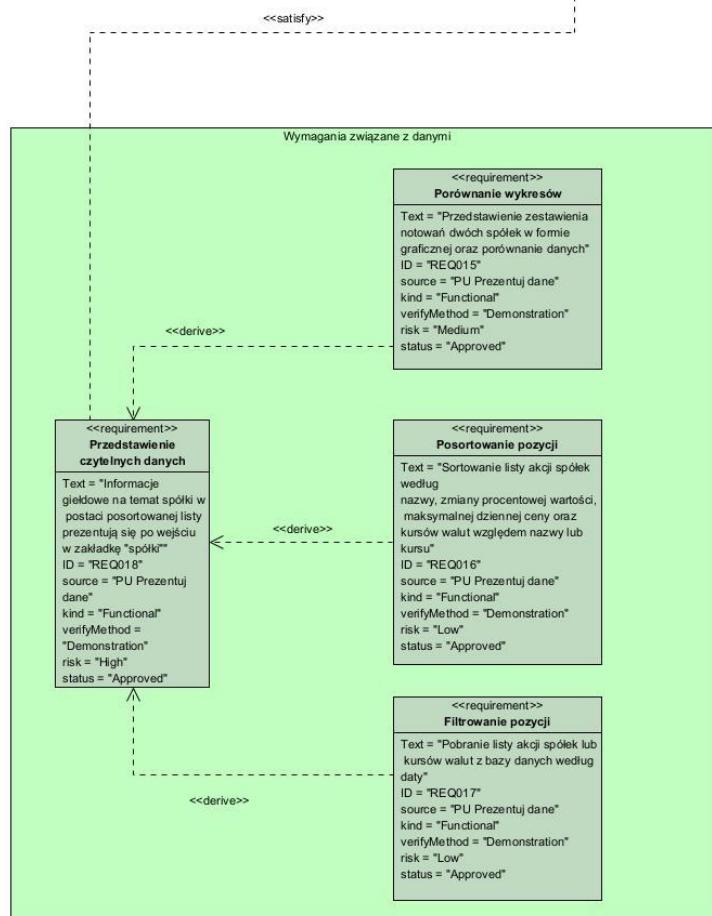
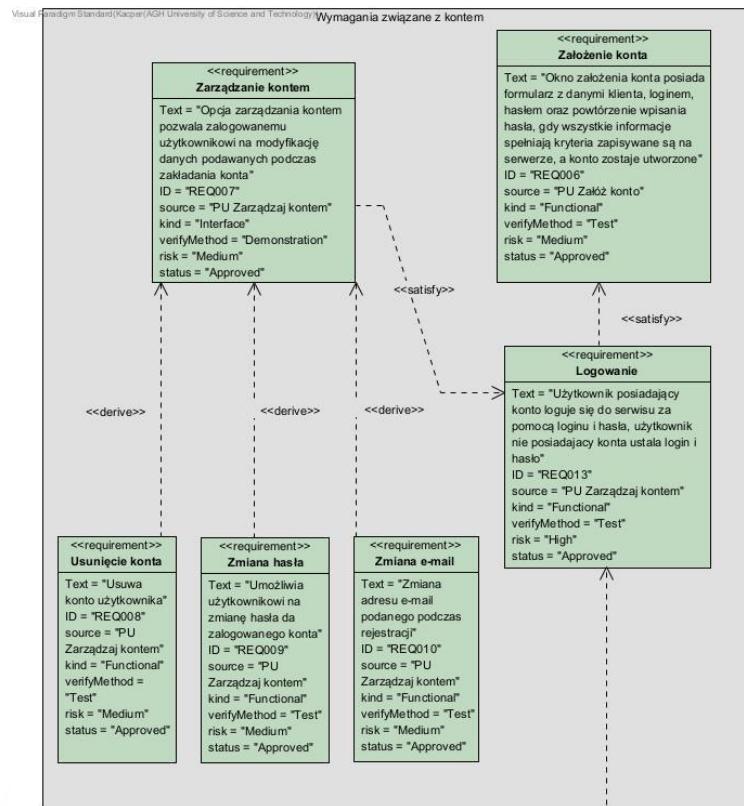
## 4. Wymagania funkcjonalne i niefunkcjonalne oraz zarys wymagań biznesowych

### *Wymagania niefunkcjonalne*

- ✓ system uruchamia się w przeglądarce internetowej,
- ✓ istnieje baza danych przechowująca dane logowania klientów, system zapewnia bezpieczeństwo danych klienta
- ✓ system pobiera z sieci rzeczywiste dane dotyczące notowań akcji spółek oraz kursów walut oraz umieszcza je w bazie danych
- ✓ system aktualizuje dane zapewniając ich poprawność
- ✓ system wyświetla pobrane dane w odpowiedniej formie



## Wymagania funkcjonalne



## *Wymagania biznesowe – ogólny zarys*

Wymagania biznesowe

**1. Skrócenie czasu potrzebnego na zebranie informacji rynkowych**

**2. Wspomaganie inwestycji dzięki prezentacji danych rynkowych w przystępny sposób**

**3. Umożliwienie podglądu danych archiwalnych**

**4. Ułatwienie porównywania notowań różnych spółek**

## 5. Diagram wymagań

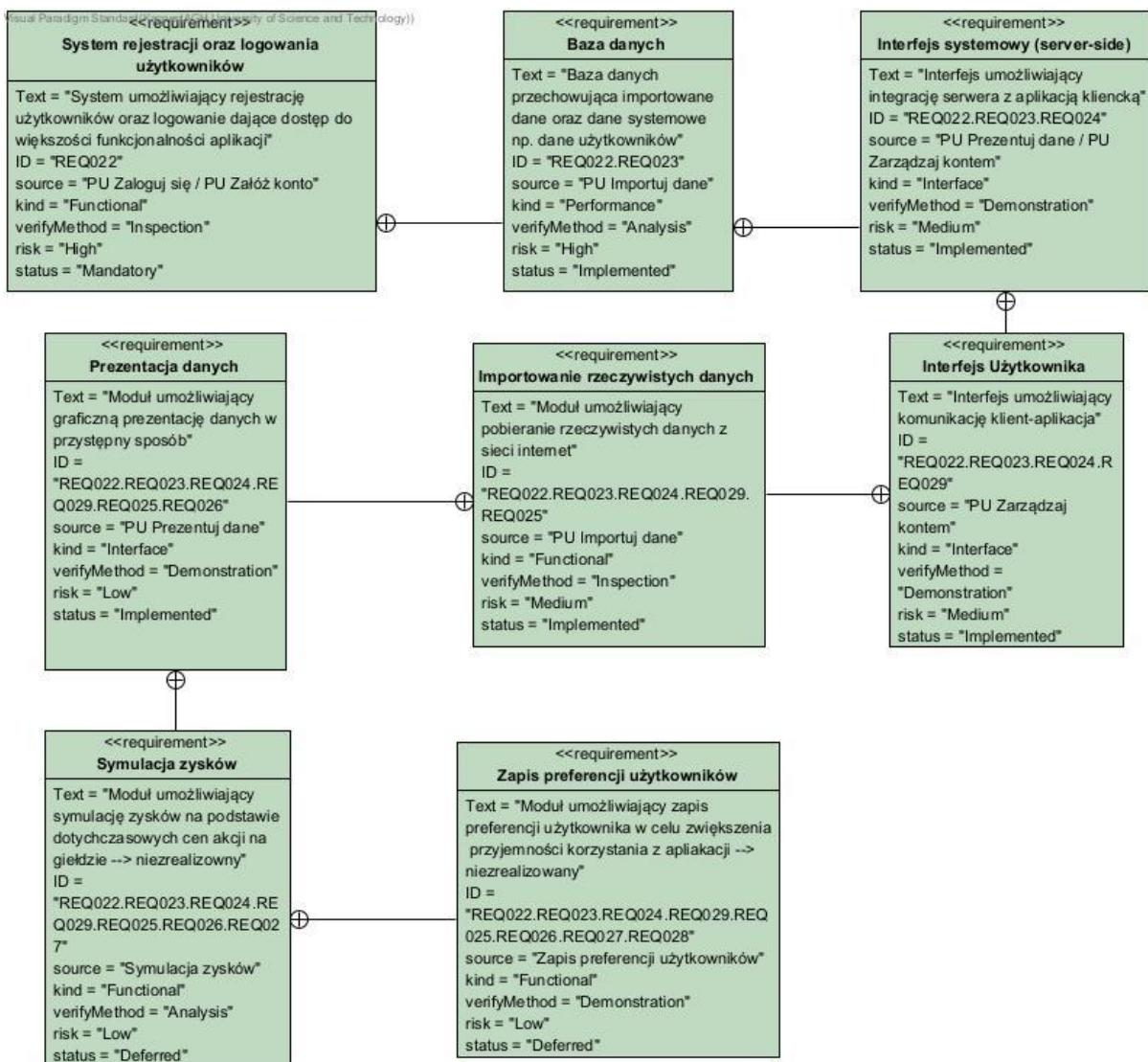


Diagram przedstawia ogólny zarys wymagań aplikacji. Powyższe moduły składają się na całość aplikacji podzielonej na część serwerową oraz aplikacyjną. Z powyższego diagramu z powodu braku zasobów w postaci doświadczenia członków zasobów oraz czasu nie udało się zrealizować modułu symulacji zysków oraz zapisu preferencji wyborów użytkownika. Aplikacja dzięki swojej modularnej budowie jest gotowa do dalszego rozwoju umożliwiającego dodanie dodatkowych funkcjonalności.

## 6. Wykaz zastosowanych framework'ów, bibliotek, środowisk oraz innych technologii i rozwiązań informatycznych



WebStorm – zintegrowane środowisko programistyczne do programowania webowego obsługujące JavaScript, HTML, CSS



Wieloplatformowe środowisko uruchomieniowe o otwartym kodzie do tworzenia aplikacji typu server-side napisanych w języku JavaScrpit

# Express

Back-endowy framework aplikacji internetowych do budowania interfejsów API z Node.js



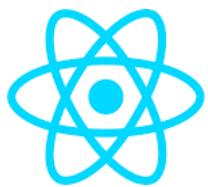
Otwarty, nierelacyjny system zarządzania bazą danych o braku ściśle zdefiniowanej struktury dokumentów składowanych w stylu JSON



Otwarty standard definujący sposób wymiany danych za pośrednictwem dokumentów JSON

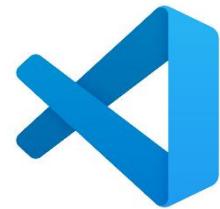


Kompatybilny z Express middleware autentyfikacyjny dla aplikacji Node.js



# React

Otwarty, nierelacyjny system zarządzania bazą danych o braku ścisłe zdefiniowanej struktury dokumentów składowanych w stylu JSON



Visual Studio Code to oferujący olbrzymie możliwości, a jednocześnie lekki i darmowy edytor programistyczny.



Hostingowy serwis internetowy przeznaczony do projektów programistycznych wykorzystujących system kontroli wersji Git.



# git

GIT to rozproszony system kontroli wersji opptymalizujący proces powstawania aplikacji, stron internetowych czy innych

```

11  "dependencies": {
12    "axios": "^1.2.2",
13    "body-parser": "^1.18.3",
14    "cors": "^2.8.5",
15    "dotenv": "^16.0.3",
16    "express": "^4.16.3",
17    "jest": "^29.3.1",
18    "jsonwebtoken": "^9.0.0",
19    "mongoose": "^5.1.2",
20    "mongoose-url-slugs": "^1.0.2",
21    "nodemailer": "^6.9.0",
22    "npm-run-all": "^4.1.5",
23    "passport": "^0.6.0",
24    "passport-jwt": "^4.0.0",
25    "passport-local-mongoose": "^7.1.2",
26    "pug": "^3.0.2",
27    "sass": "^1.54.5",
28    "slugify": "^1.6.5",
29    "supertest": "^6.3.3",
30    "xlsx": "^0.18.5"
31  },
32  "devDependencies": {
33    "assert": "^2.0.0",
34    "aws-sdk": "^2.1283.0",
35    "browserify-zlib": "^0.2.0",
36    "buffer": "^5.7.1",
37    "crypto-browserify": "^3.12.0",
38    "events": "^3.3.0",
39    "os-browserify": "^0.3.0",
40    "path-browserify": "^1.0.1",
41    "process": "^0.11.10",
42    "punycode": "^1.4.1",
43    "querystring-es3": "^0.2.1",
44    "stream-browserify": "^3.0.0",
45    "stream-http": "^3.2.0",
46    "string_decoder": "^1.3.0",
47    "url": "^0.11.0",
48    "util": "^0.12.5"
49  }

```

<sup>31</sup> Zawartość pliku package.json definiującego zależności części backend-owej projektu, technologie i biblioteki wykorzystywane na serwerze

```

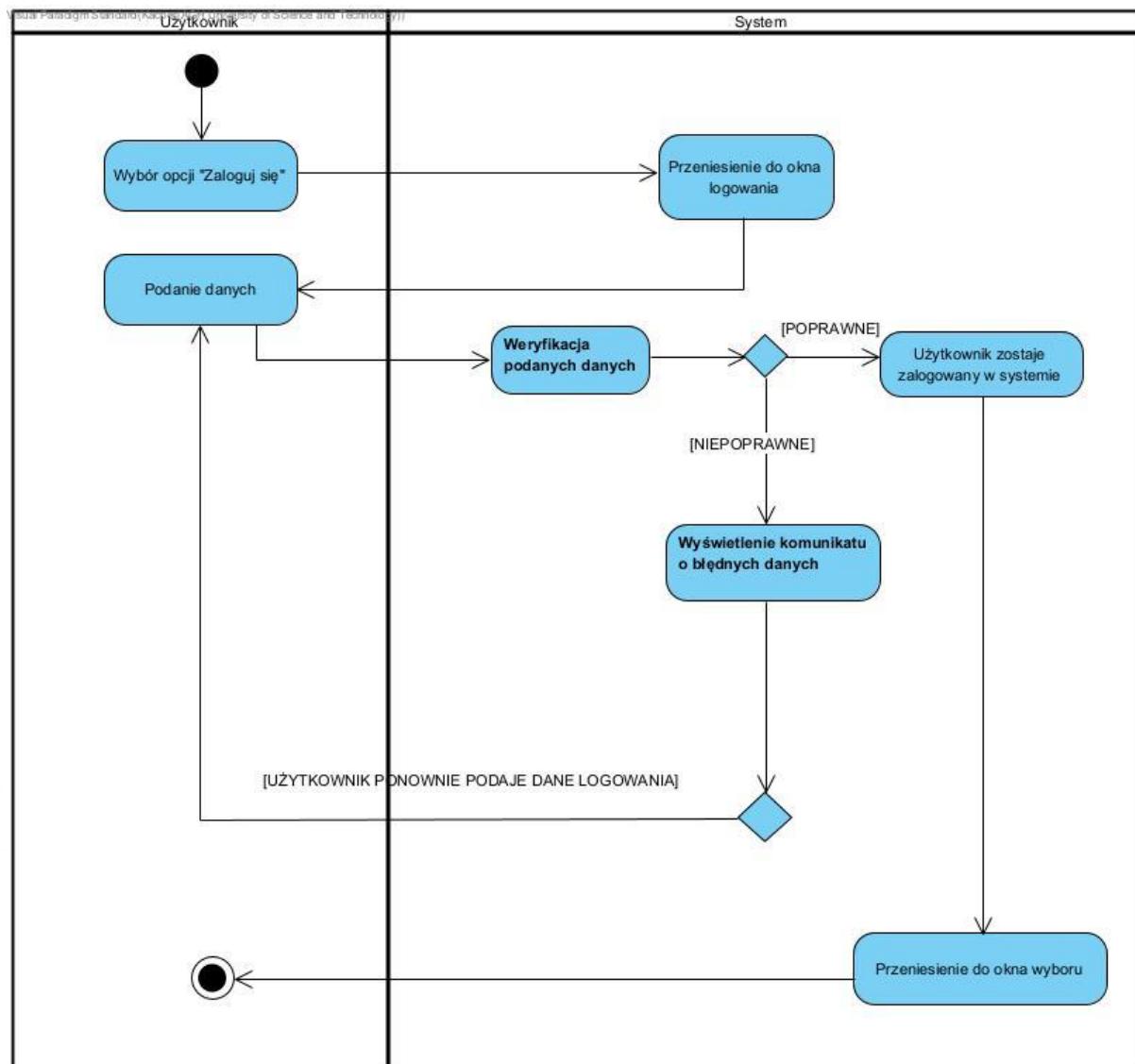
5   "dependencies": {
6     "@emotion/styled": "^11.10.5",
7     "@fortawesome/fontawesome-svg-core": "^6.2.1",
8     "@fortawesome/free-solid-svg-icons": "^6.2.1",
9     "@fortawesome/react-fontawesome": "^0.2.0",
10    "@testing-library/jest-dom": "^5.16.5",
11    "@testing-library/react": "^13.4.0",
12    "@testing-library/user-event": "^13.5.0",
13    "axios": "^1.2.2",
14    "react": "^18.2.0",
15    "react-bootstrap": "^2.7.0",
16    "react-dom": "^18.2.0",
17    "react-router-dom": "^6.6.2",
18    "react-scripts": "5.0.1",
19    "react-table": "^7.8.0",
20    "styled-component": "^2.8.0",
21    "twin.macro": "^3.1.0",
22    "web-vitals": "^2.1.4",
23    "jwt-decode": "^3.1.2"
24  },

```

Zawartość pliku package.json definiującego zależności części frontend-owej projektu, technologie i biblioteki wykorzystywane na serwerze

## 7. Diagramy aktywności

Diagram aktywności dla przypadku użycia „Zaloguj się”



### Diagram aktywności dla przypadku użycia „Załóż konto”

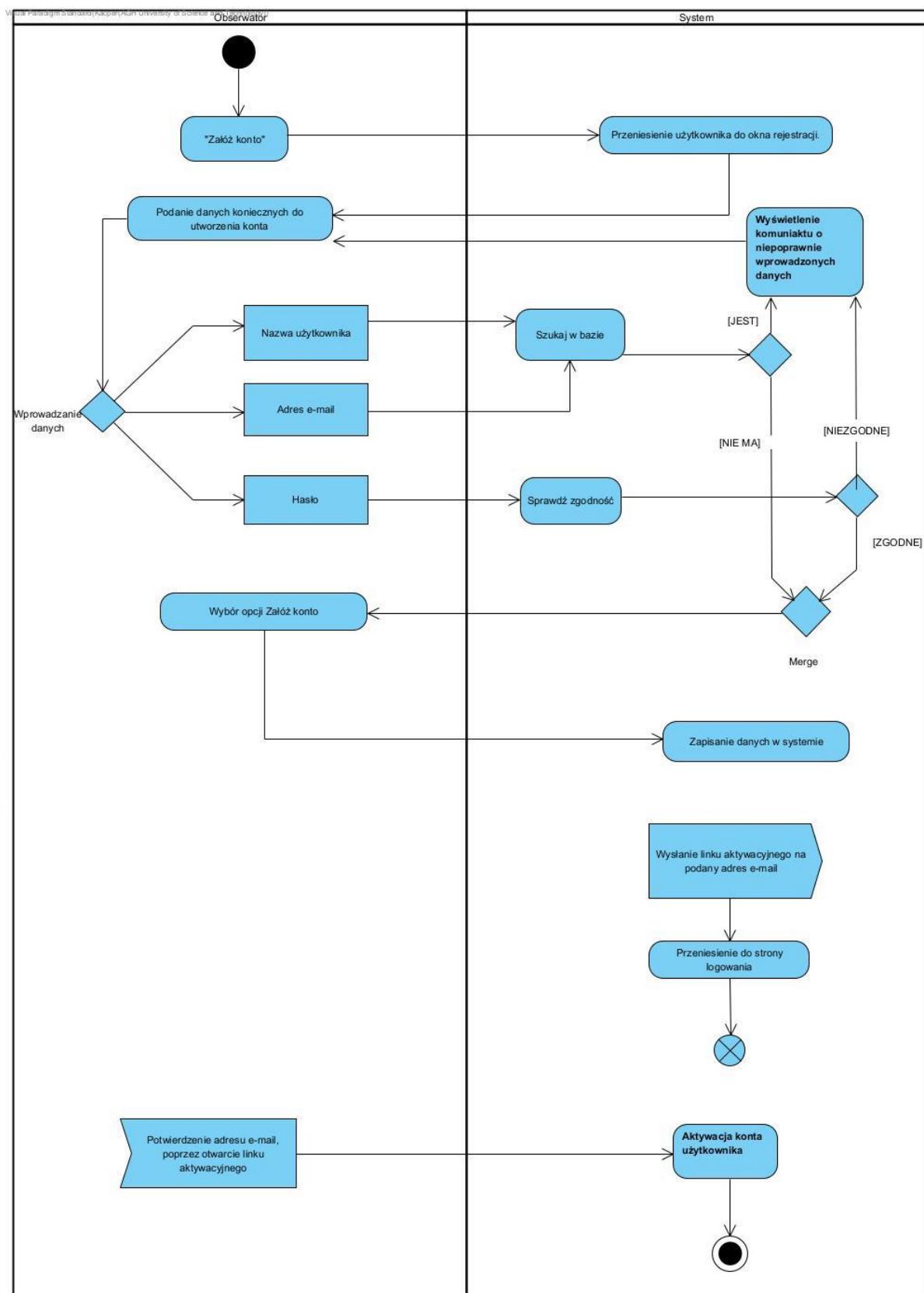


Diagram aktywności dla przypadku użycia „Importuj dane”

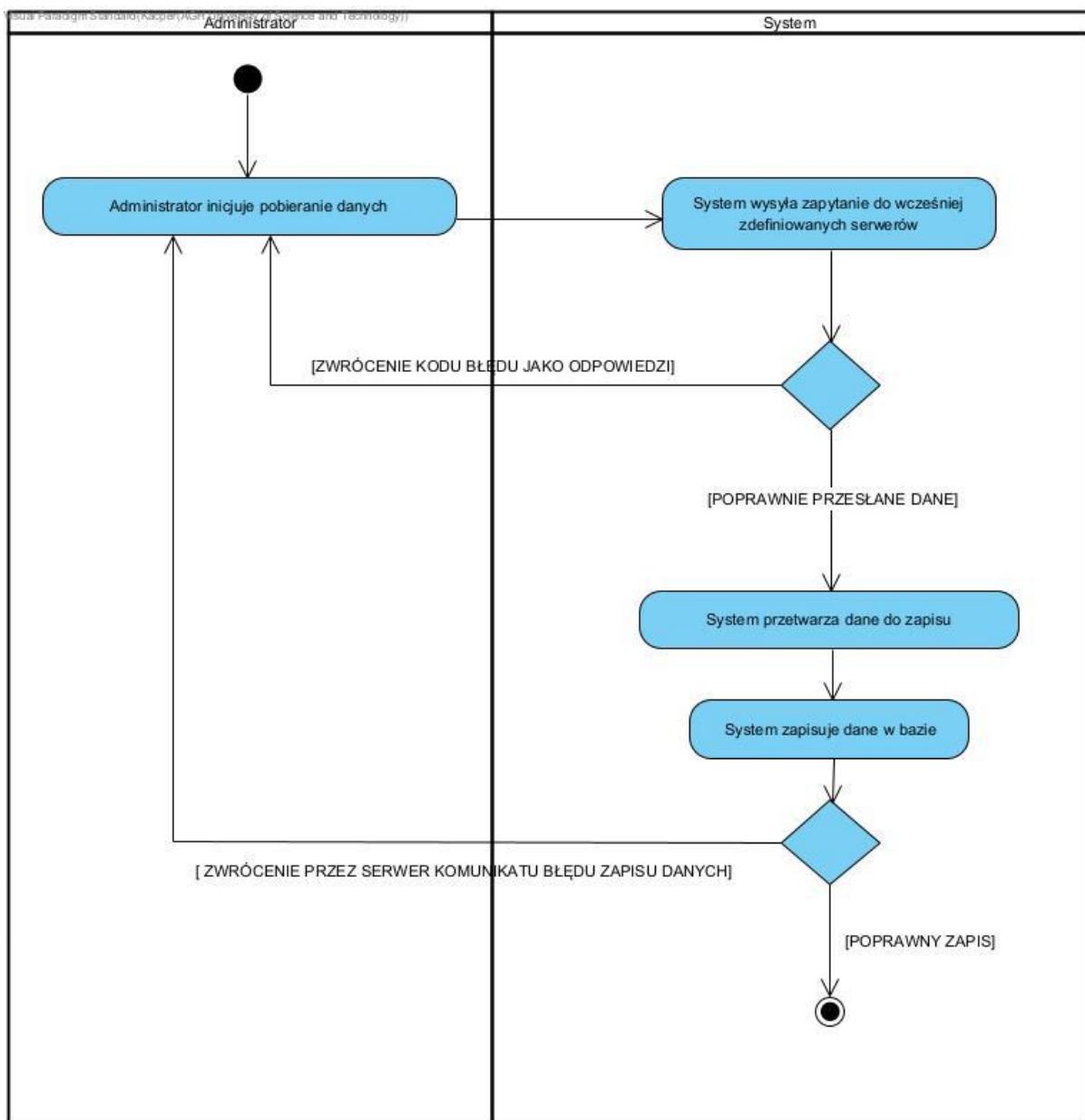
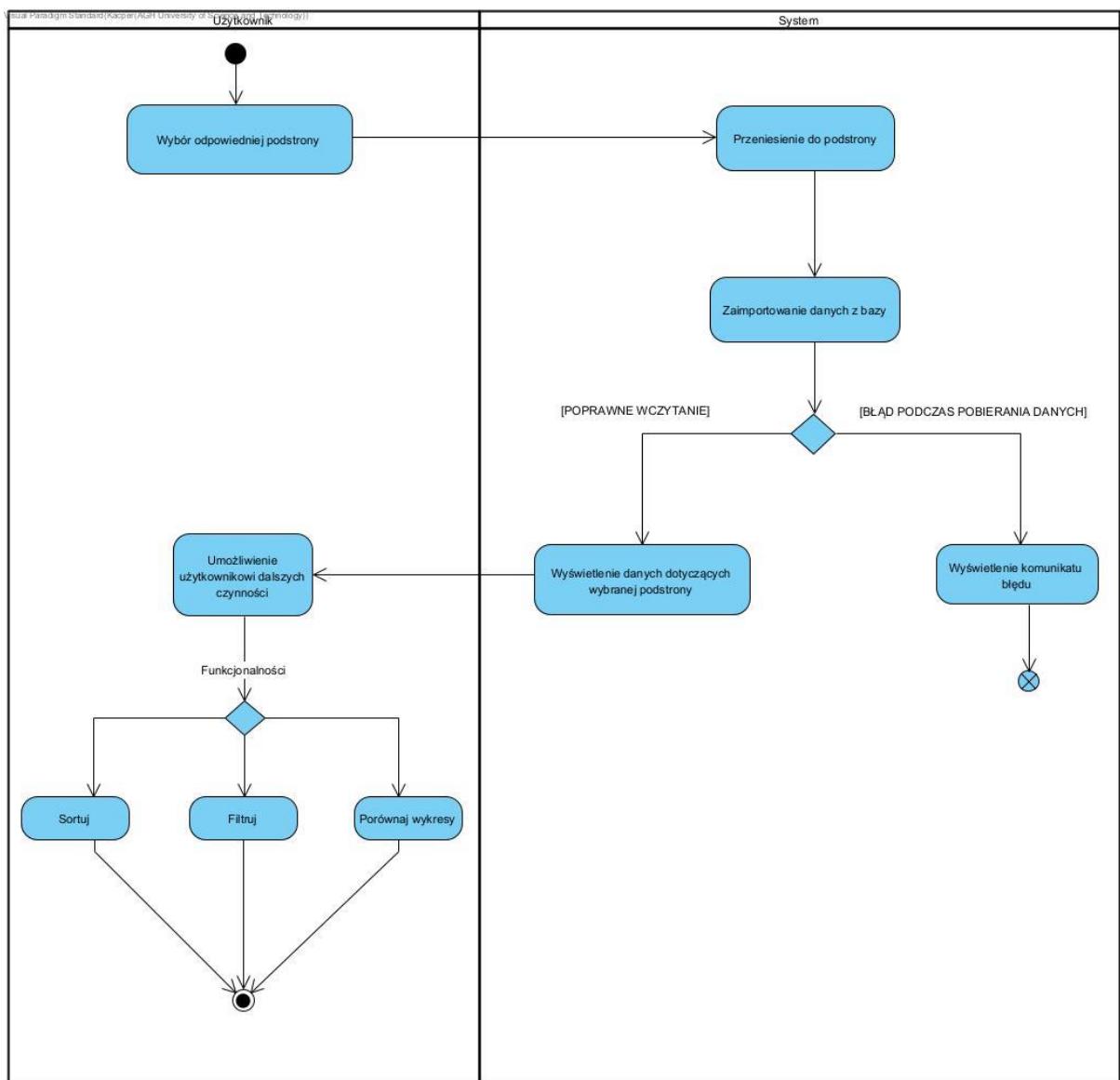
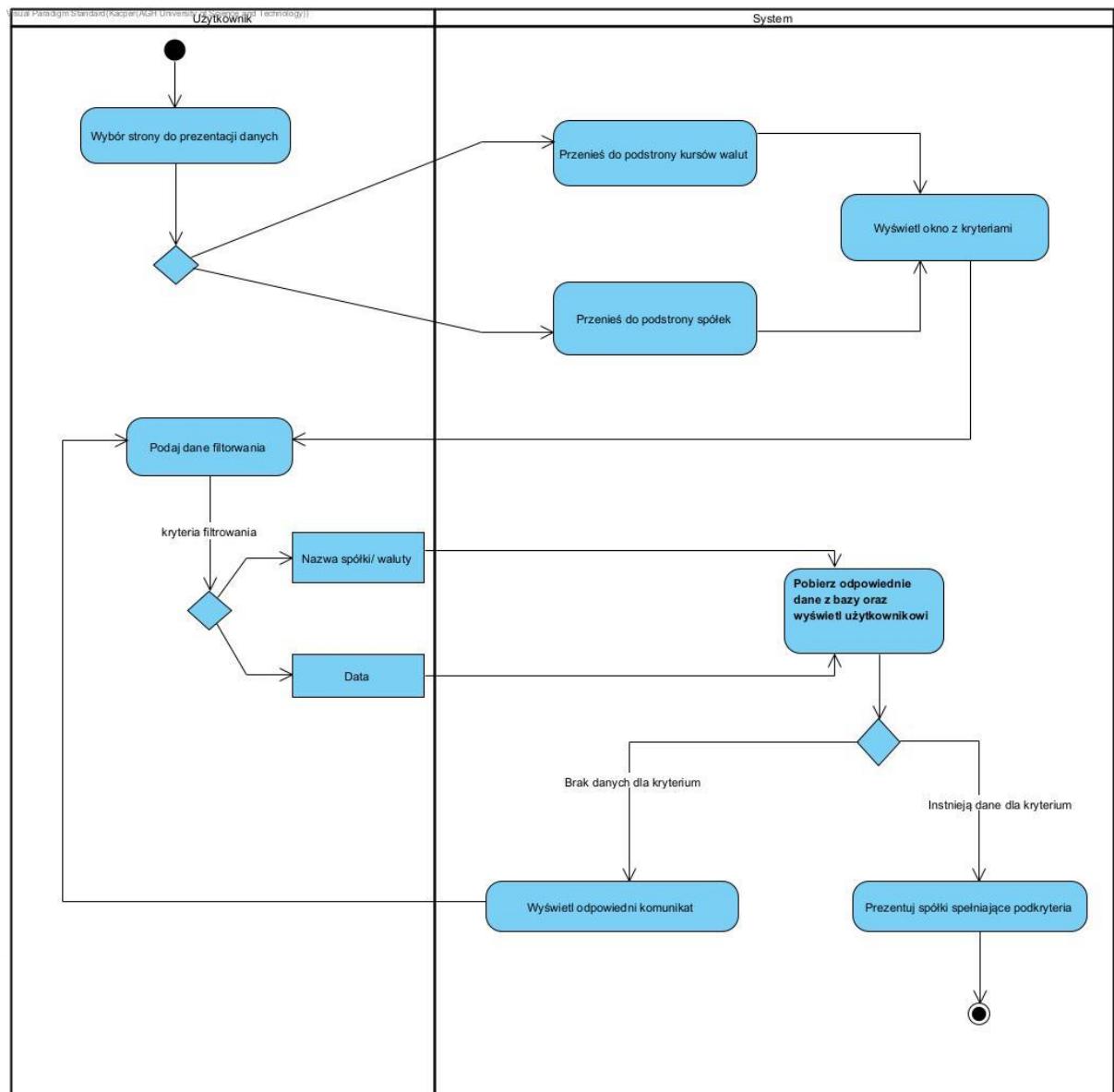


Diagram aktywności dla przypadku użycia „Prezentuj dane”



*Diagram aktywności dla przypadku użycia „Filtruj dane”*



*Diagram aktywności dla przypadku użycia „Sortuj dane”*

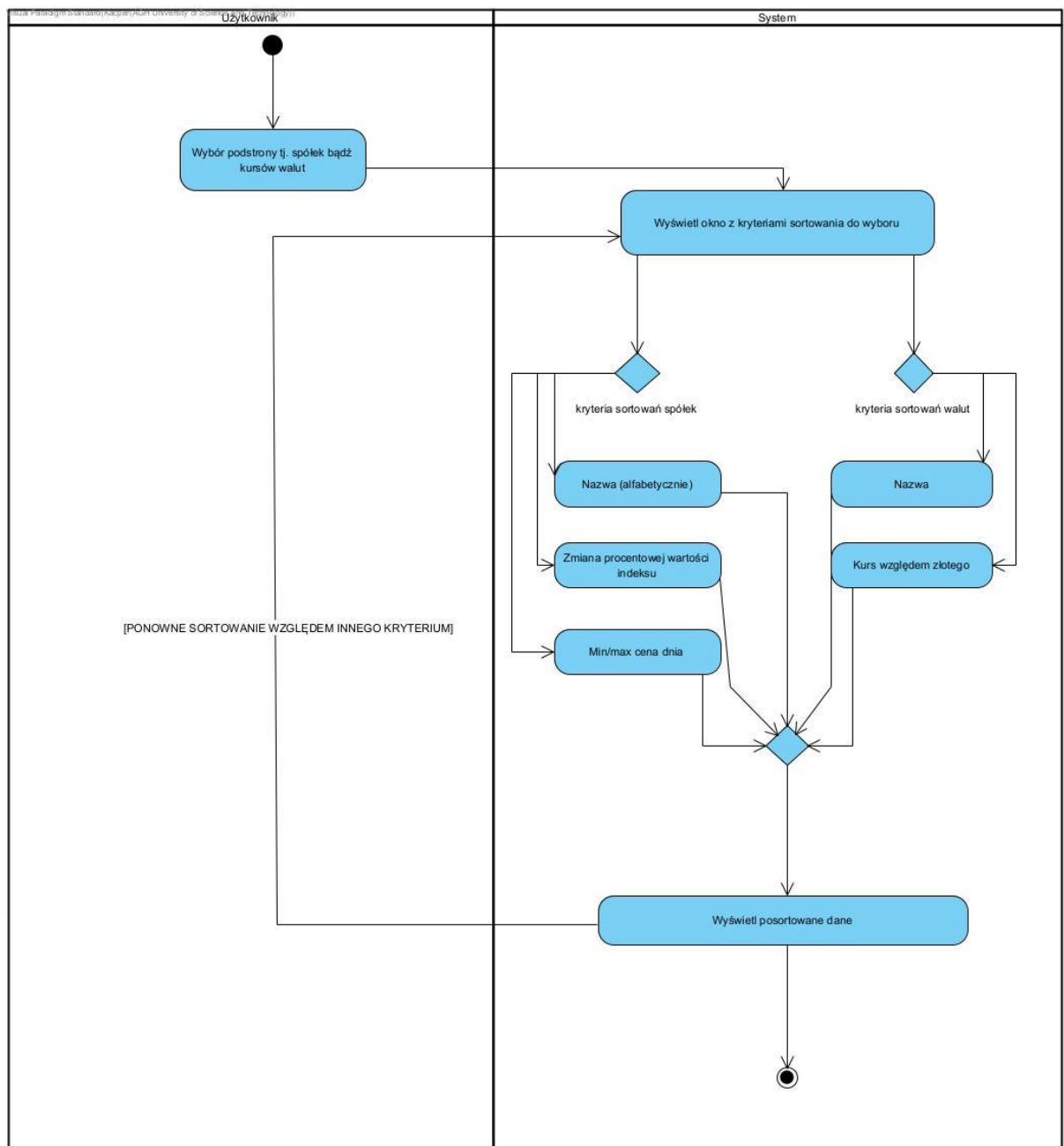
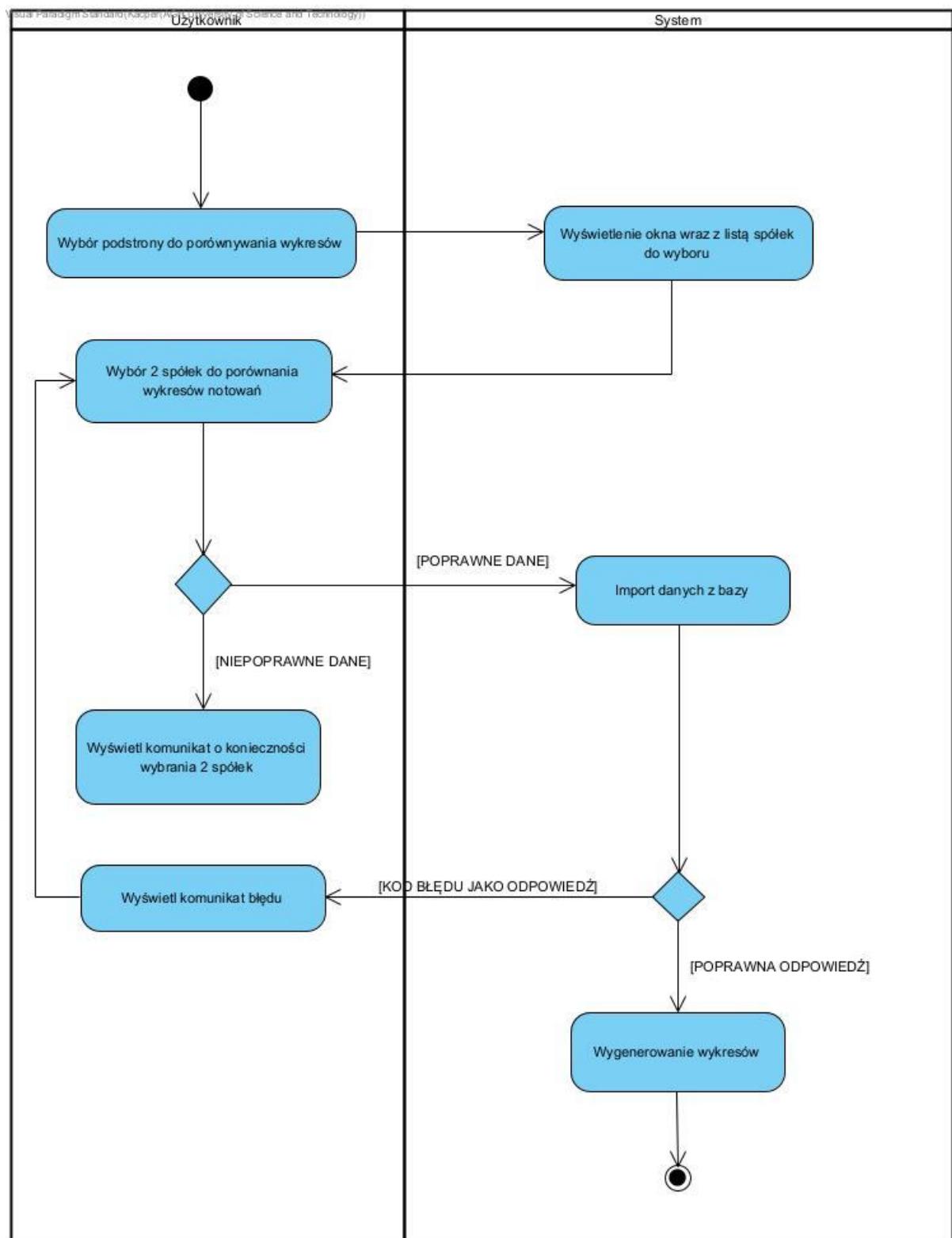


Diagram aktywności dla przypadku użycia „Porównaj wykresy”



### Diagram aktywności dla przypadku użycia „Zarządzaj serwisem”

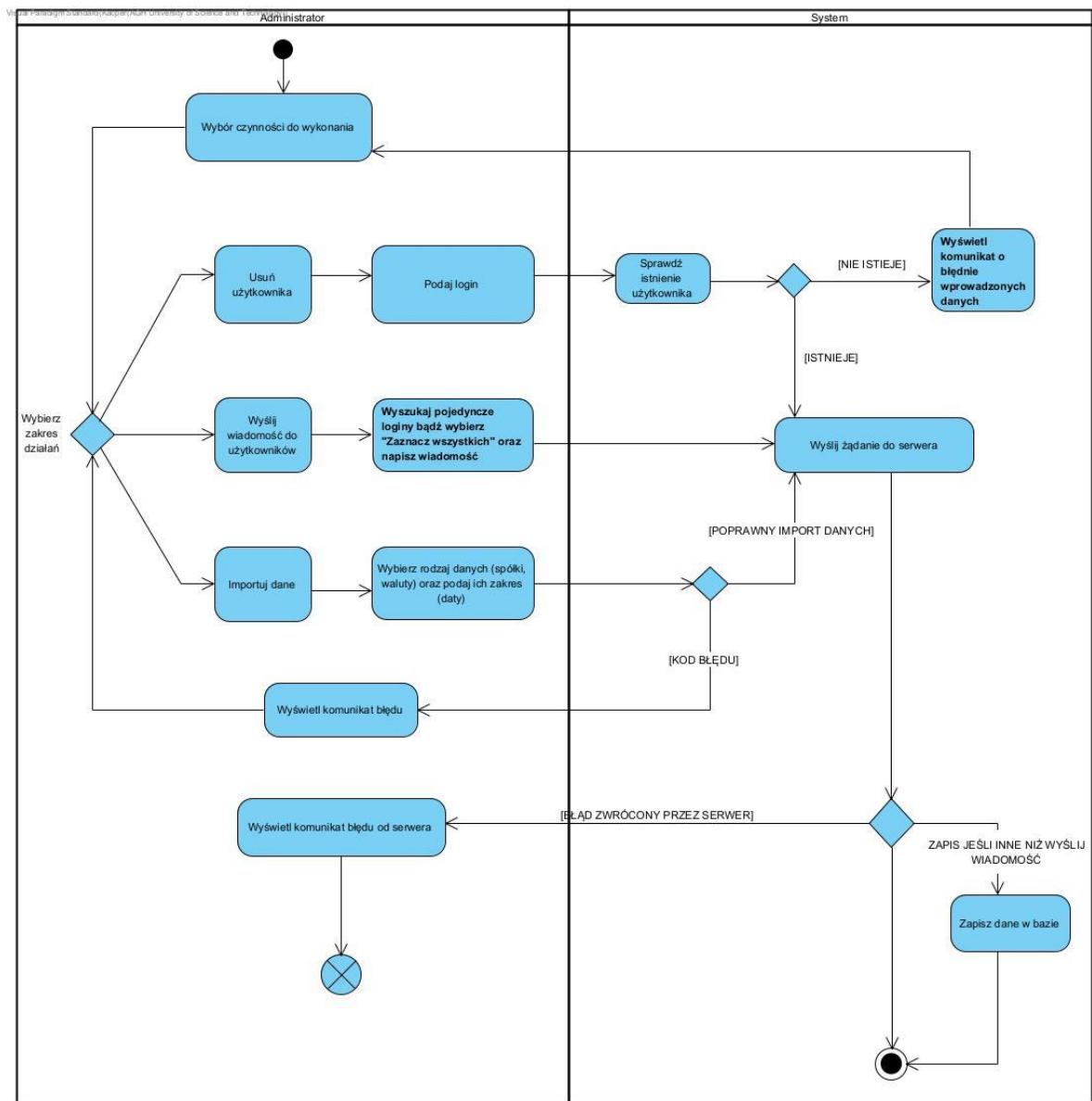
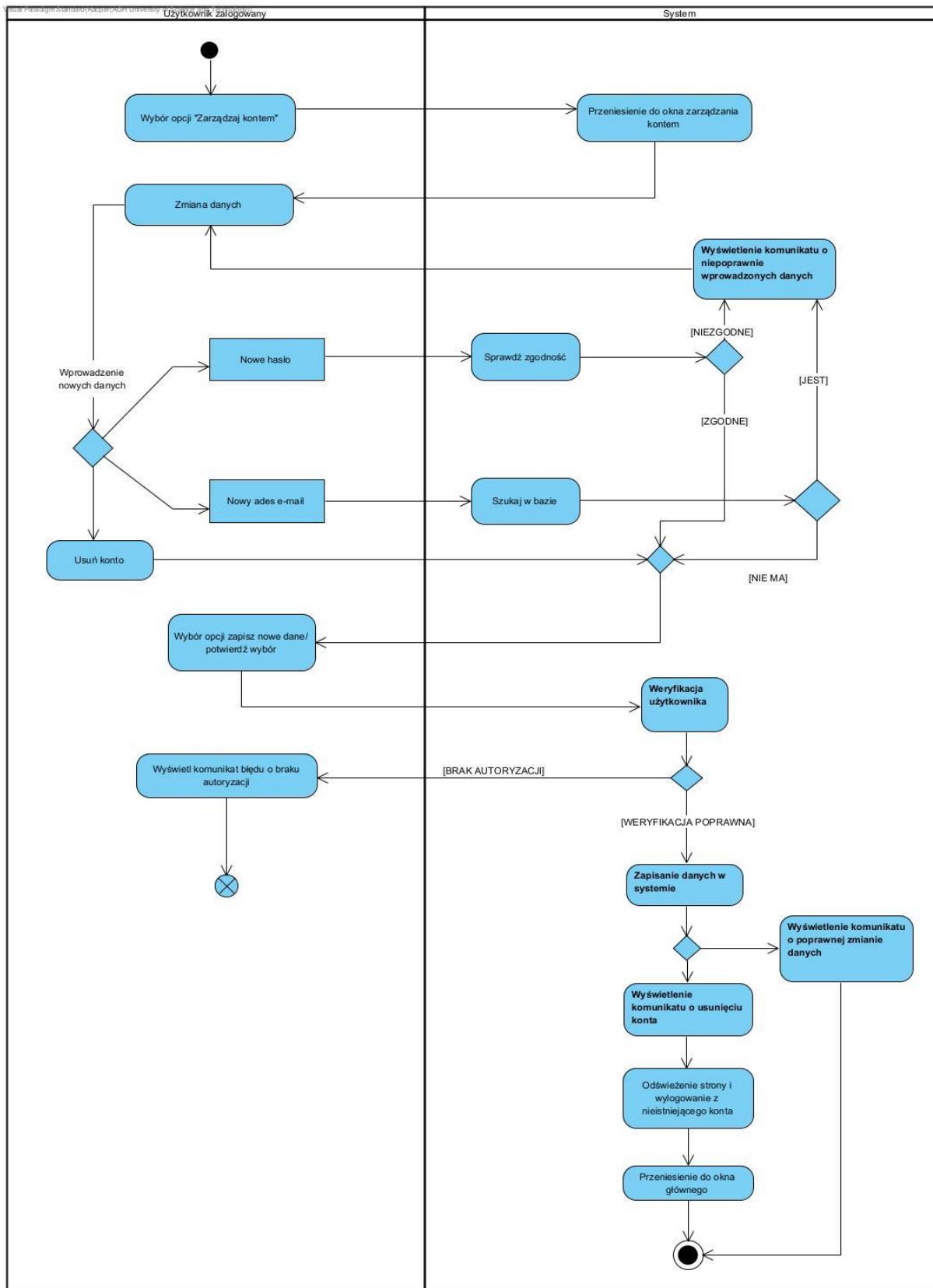


Diagram aktywności dla przypadku użycia „Zarządzaj kontem”



## 8. Diagram klas

Diagram klas części backend'owej aplikacji (serwera) opiera się na modelach stworzonych w bazie danych MongoDB. Za pośrednictwem biblioteki mongoose w projekcie implementujemy interfejsy umożliwiające manipulację tymi danymi.

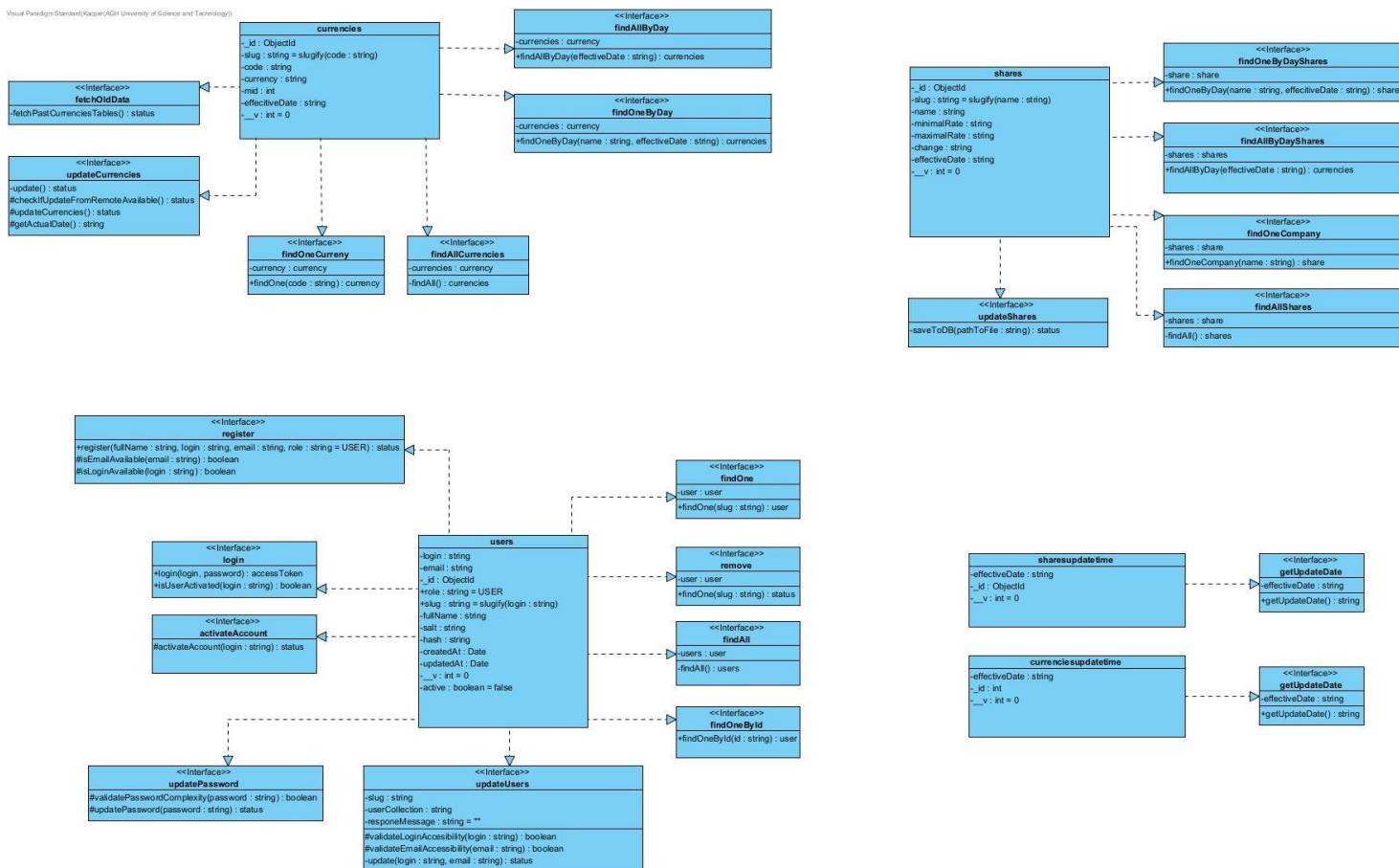
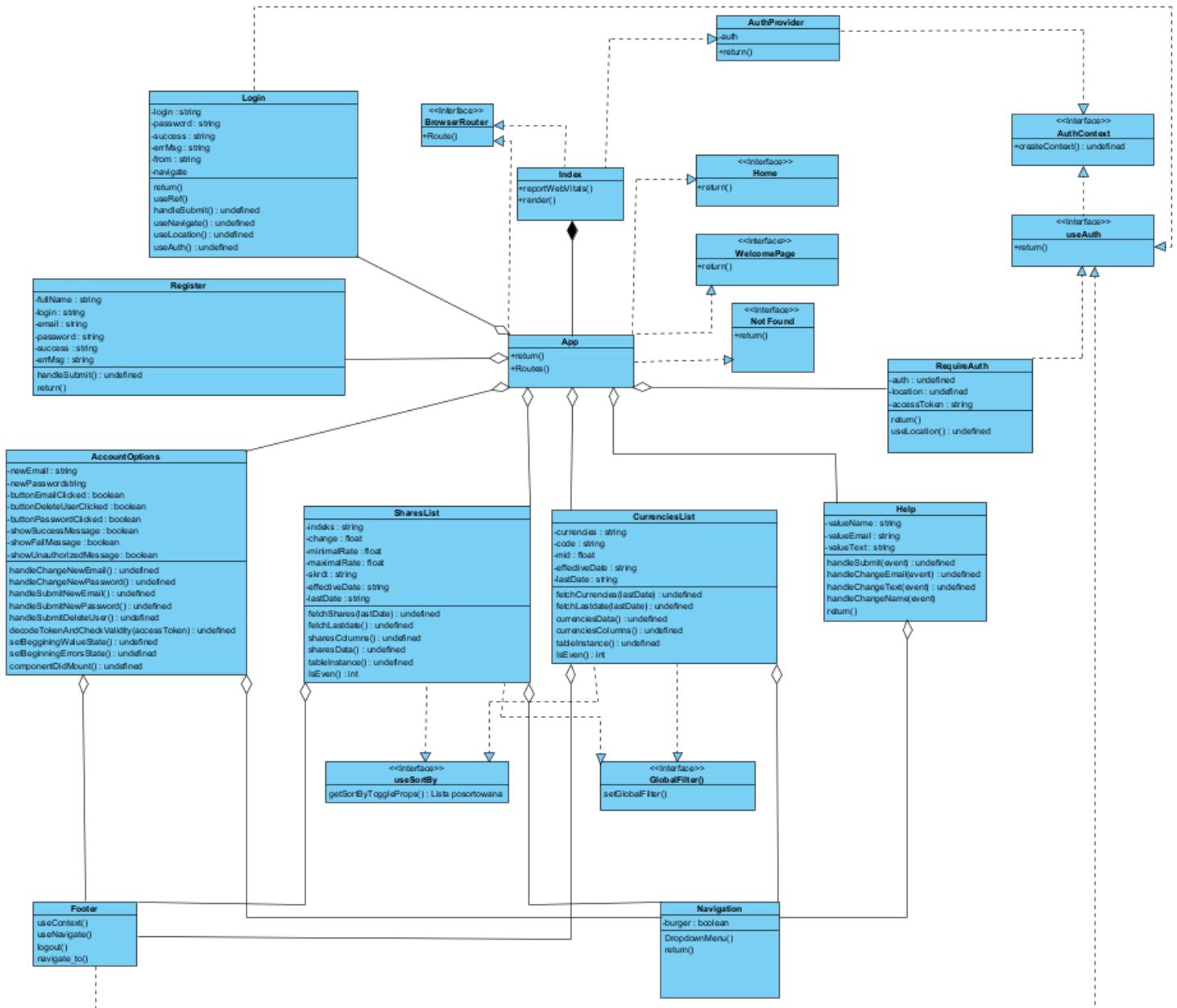
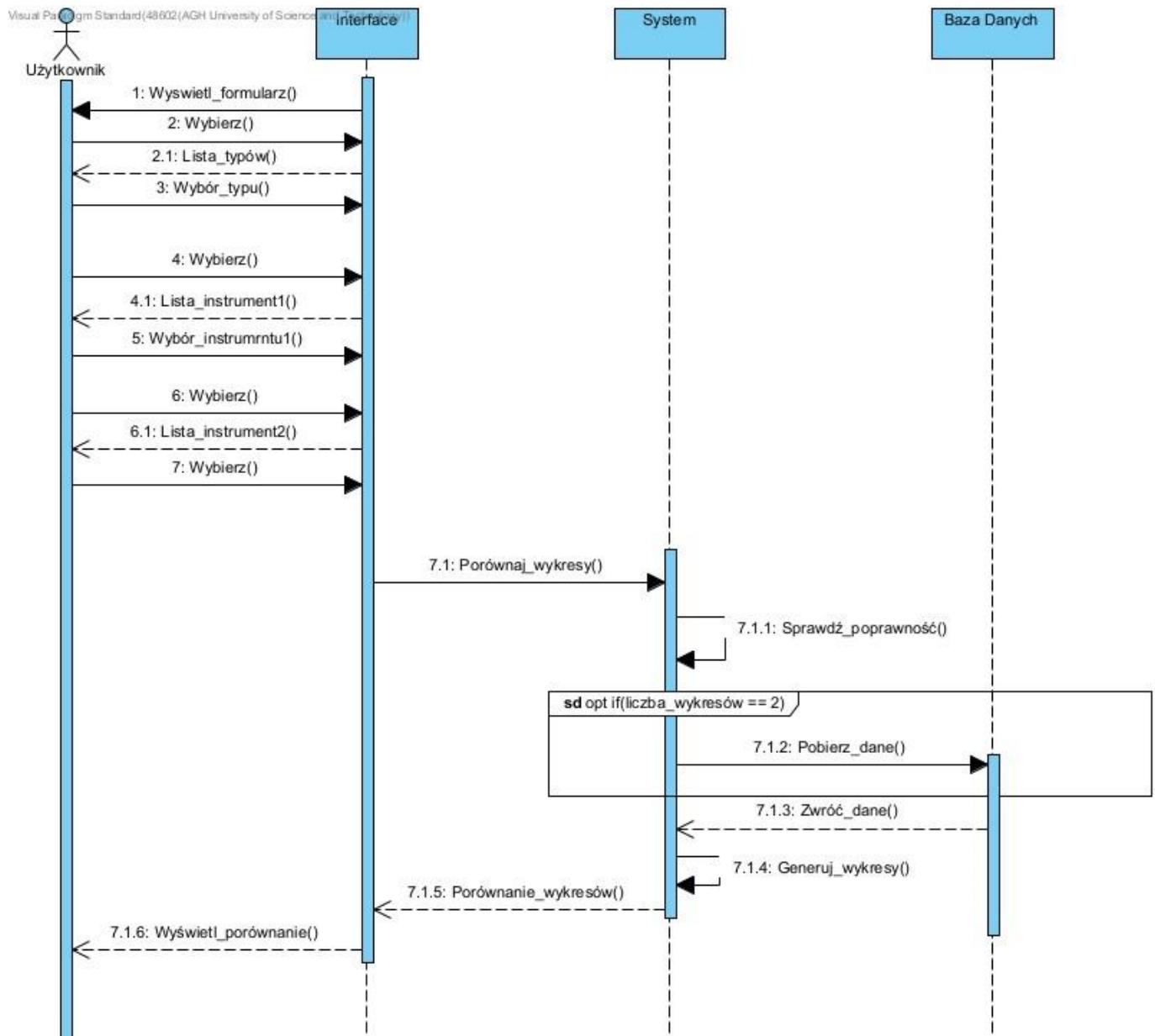


Diagram klas części frontendowej napisanej w Reakcie, oparty jest na komponentach reactowych

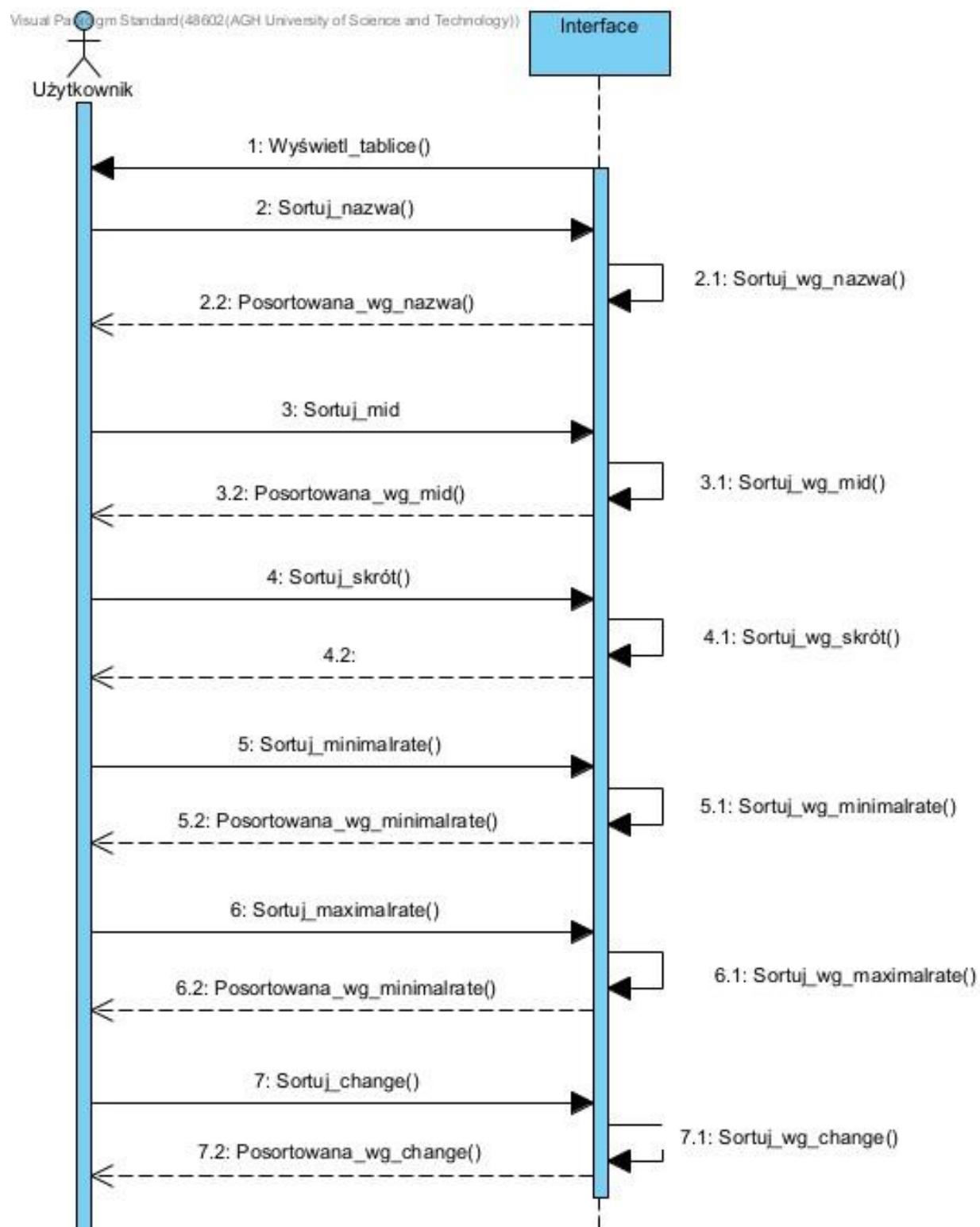


## 9. Diagram sekwencji

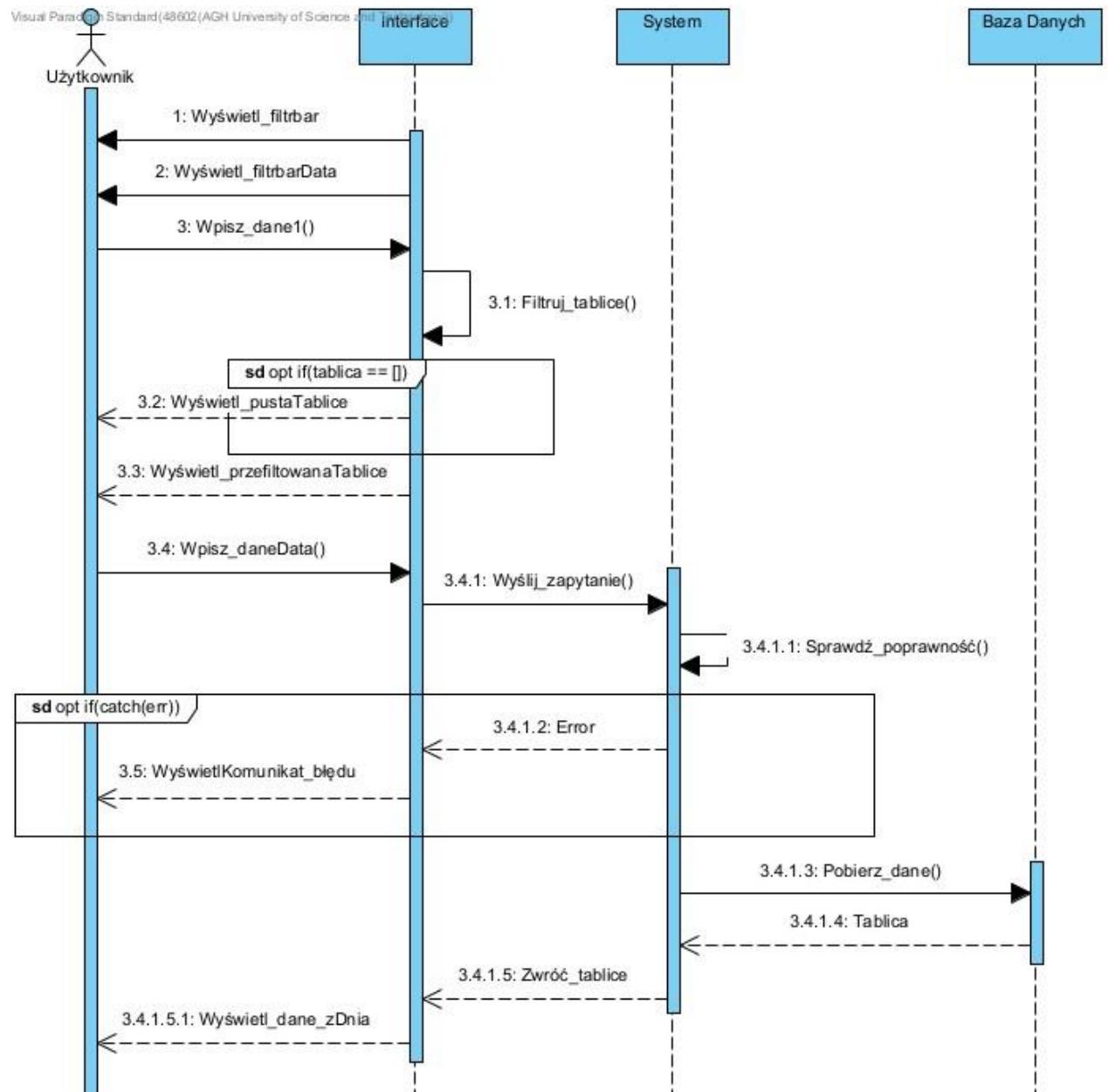
### PORÓWNAJ WYKRESY



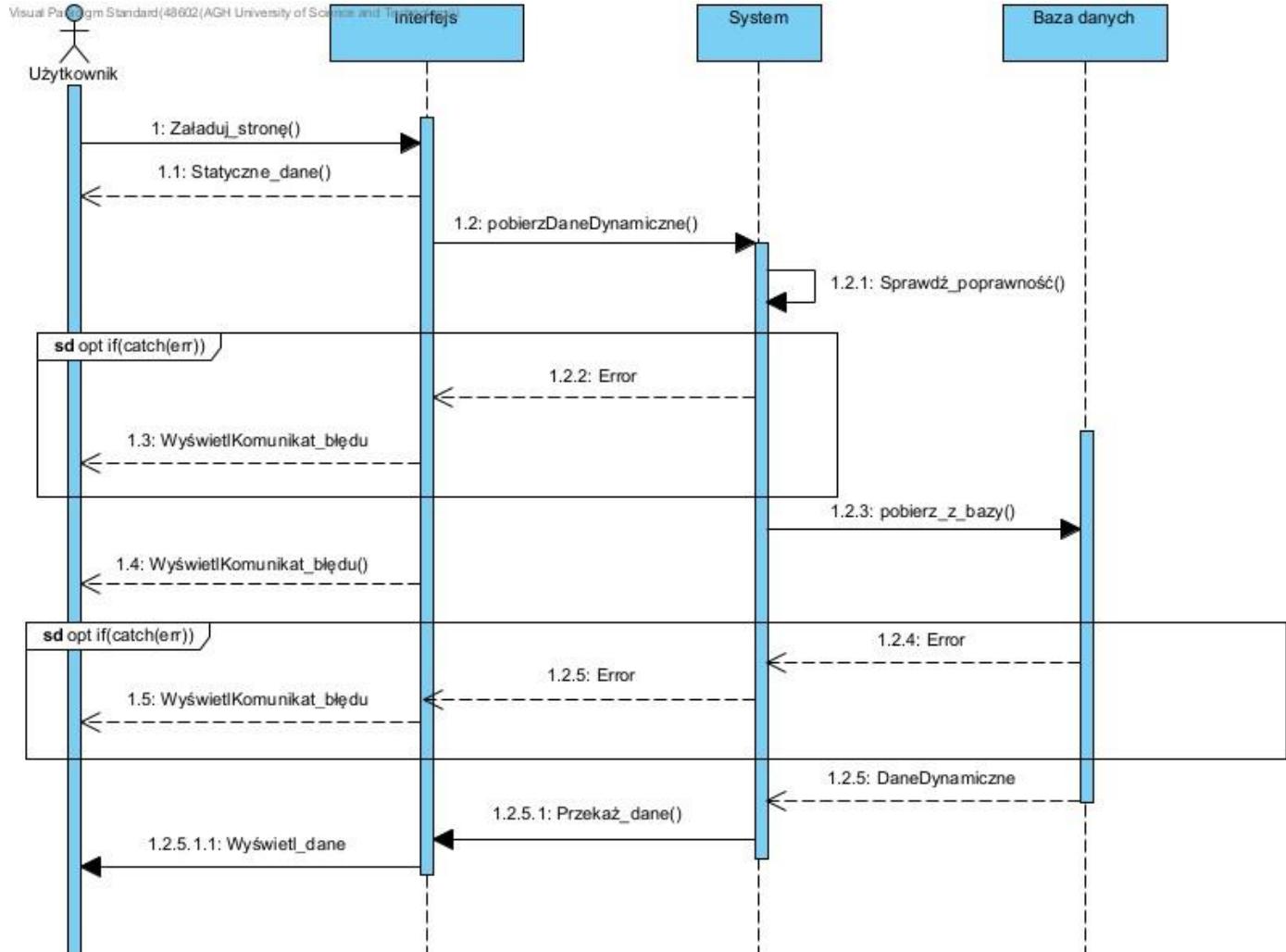
## SORTUJ DANE



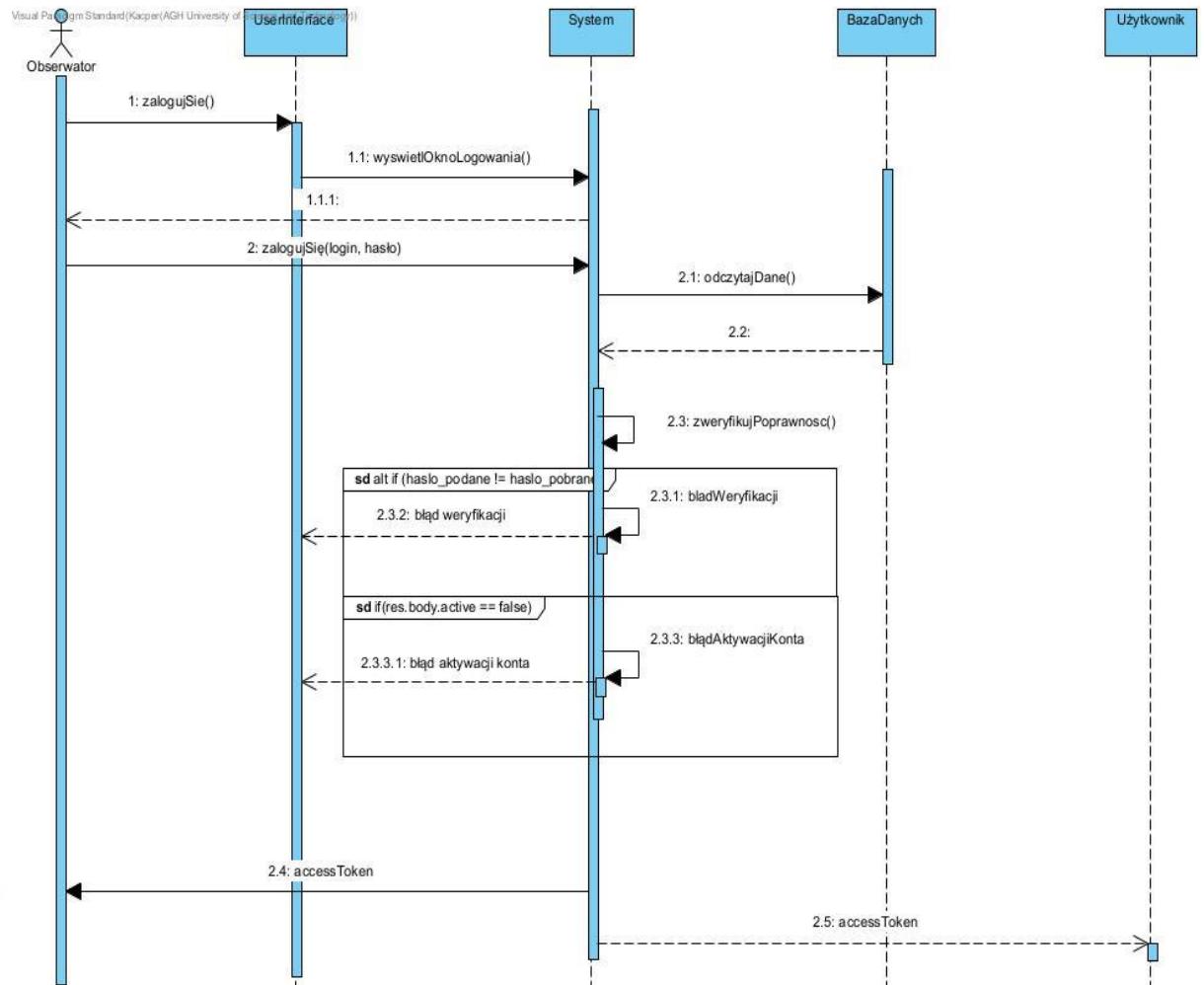
## FILTRUJ DANE



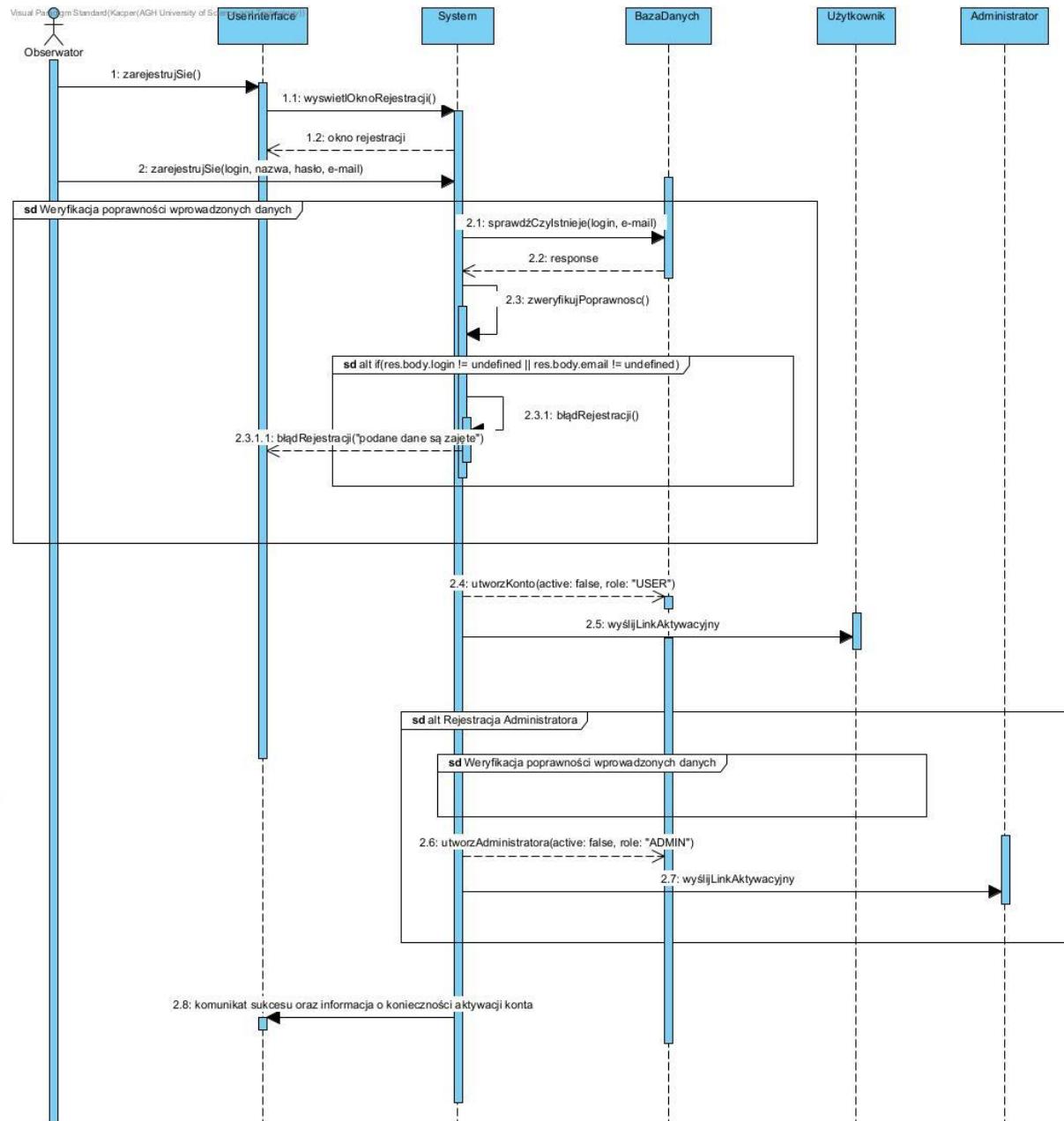
## WYŚWIETL DANE



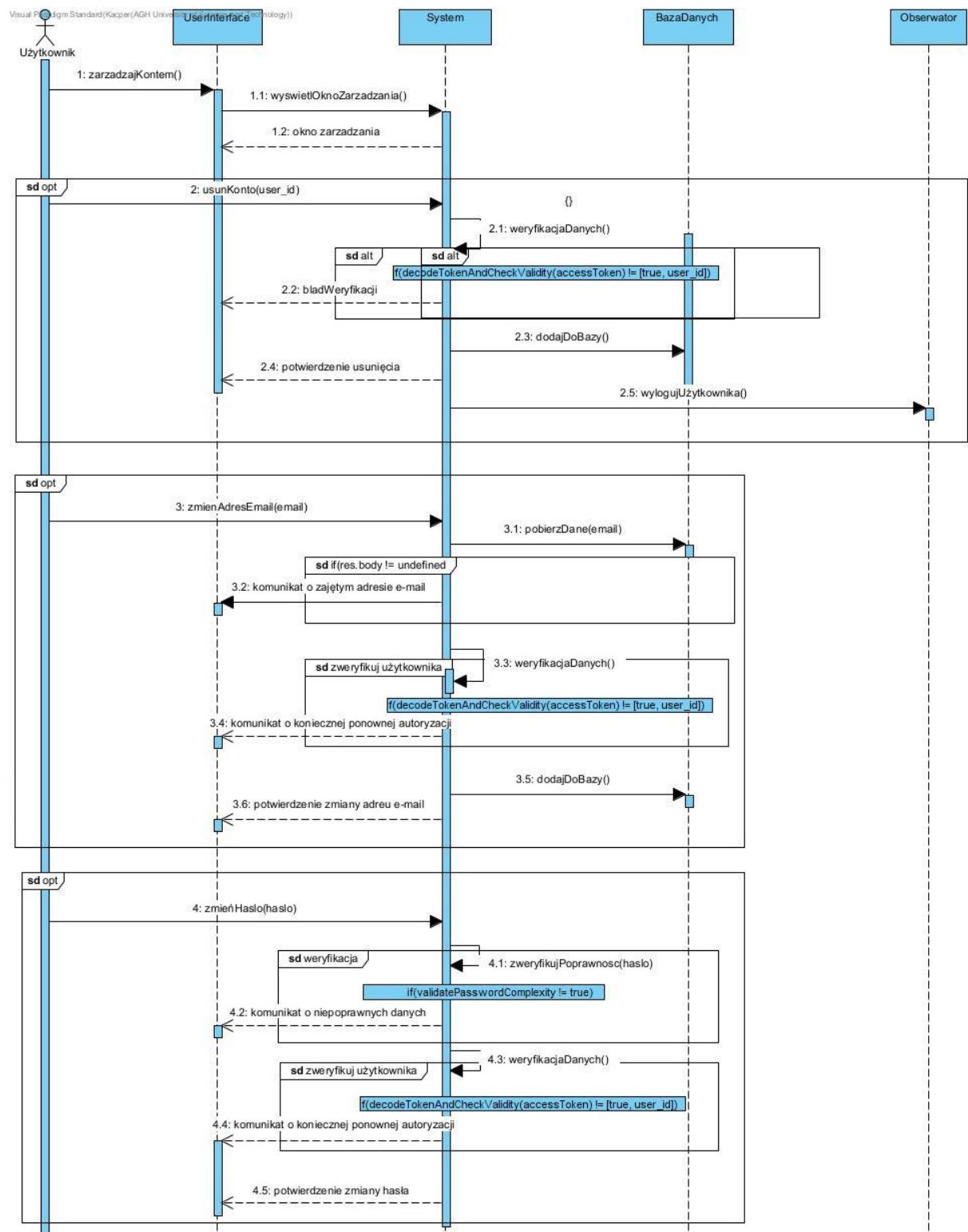
## LOGOWANIE



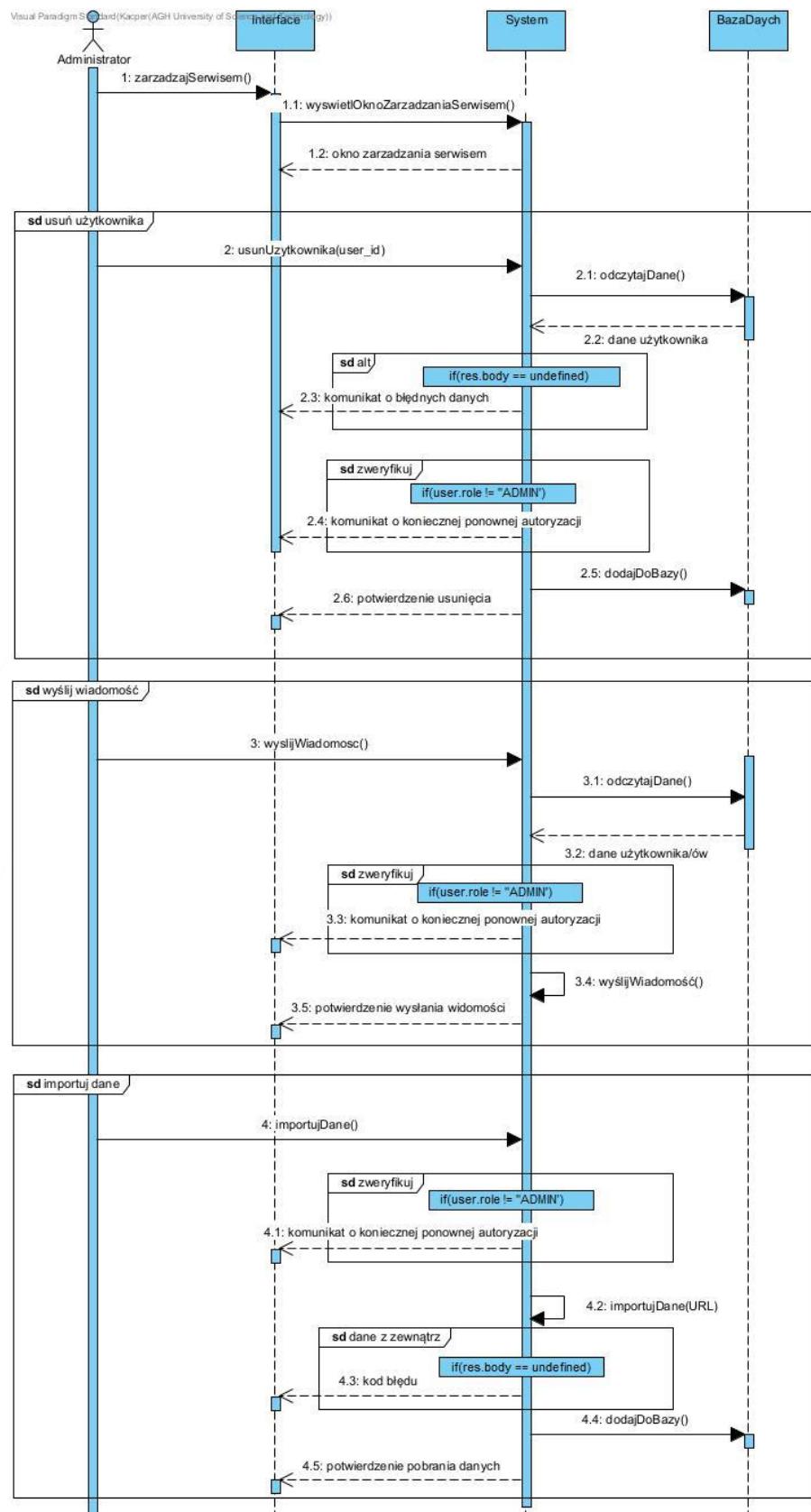
# ZAŁÓŻ KONTO



## ZARZĄDZAJ KONTEM

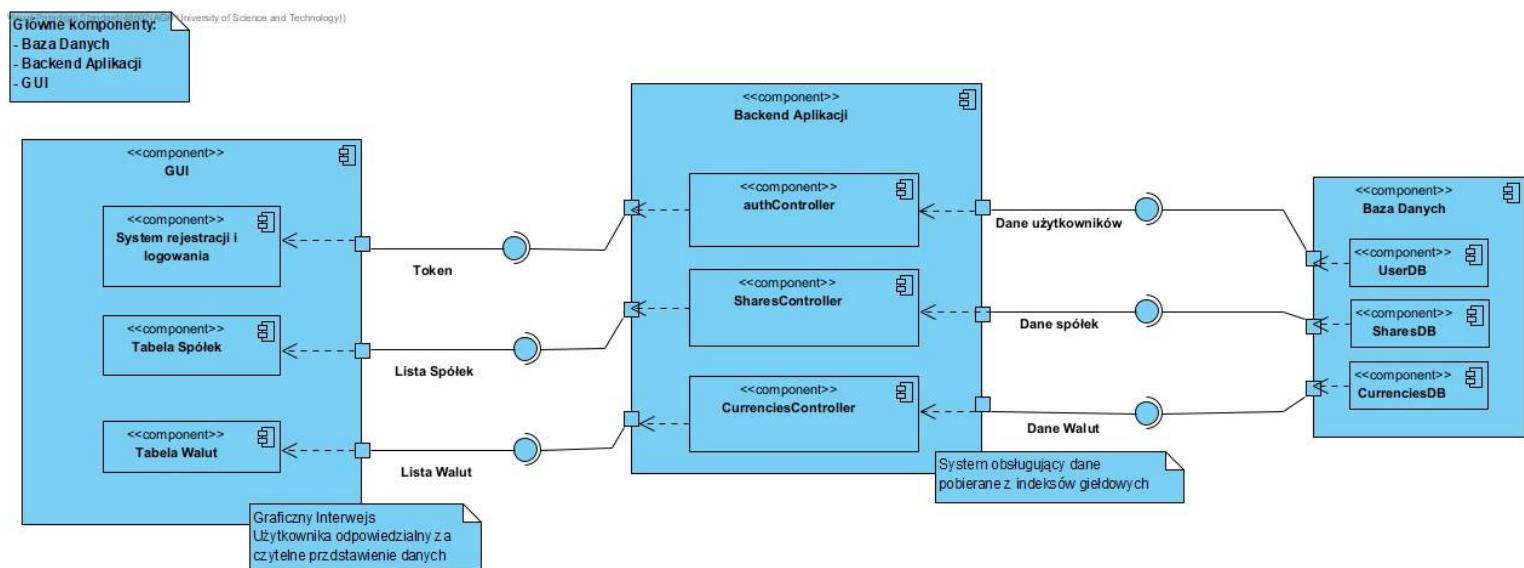


## ZARZĄDZAJ SERWISEM



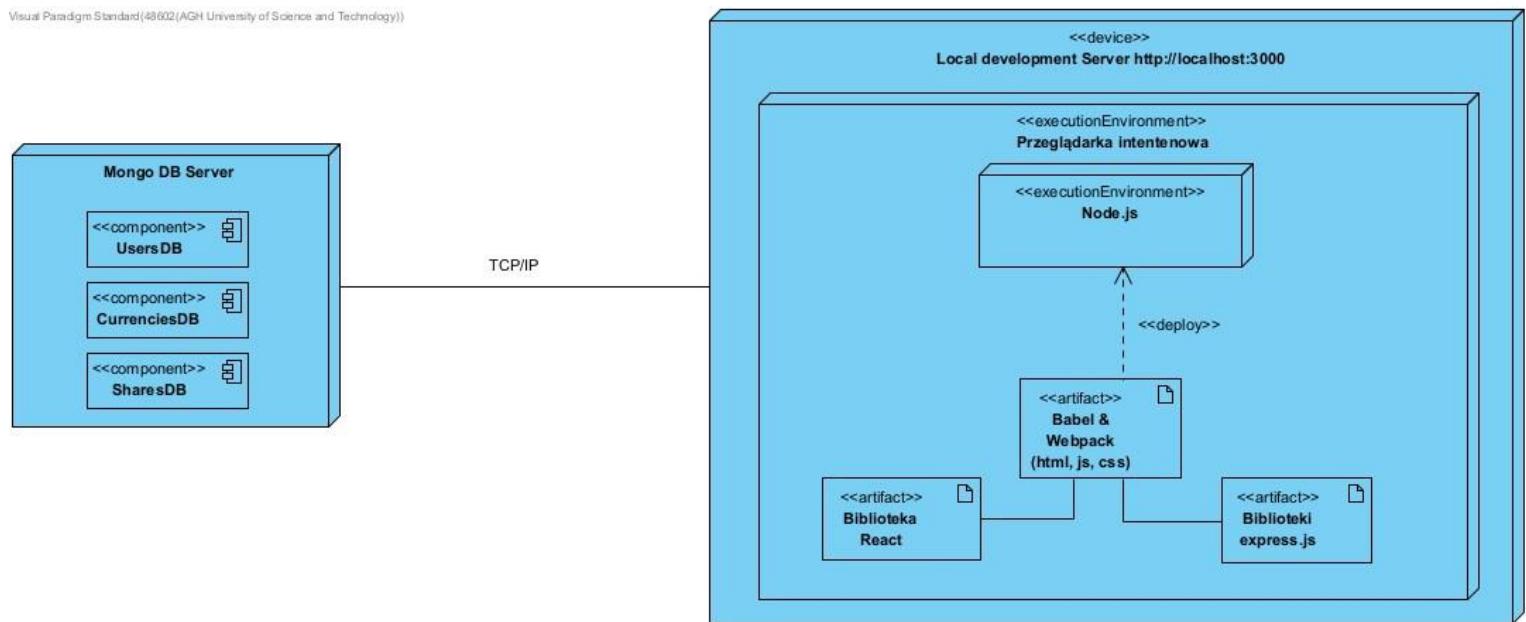
## 10. Diagram komponentów

Diagram komponentów służy do ukazania organizacji pomiędzy komponentami, zależności między głównymi częściami systemu oraz służy do określania szczegółów niezbędnych do budowy systemu. Głównym jego zadaniem jest określenie, który komponent wymaga jakiego interfejsu, dostarczanego przez inny. W naszym projekcie posiadamy trzy główne komponenty odpowiedzialne za magazynowanie danych (baza danych), ich transport oraz przetwarzanie (backend), a także za przekazanie ich użytkownikowi (frontend). Główne komponenty dzielą się na trzy mniejsze ze względu na trzy główne rodzaje danych występujące w naszym projekcie.



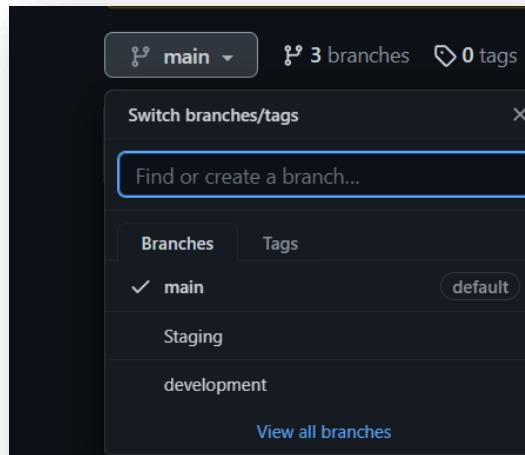
## 11. Diagram wdrożenia

Visual Paradigm Standard (48602 (AGH University of Science and Technology))



## 12. Przykłady współpracy zespołu

Jako główną platformę dla współpracy zespołu wykorzystaliśmy githuba, gdzie utworzone zostały trzy branche: main, development oraz Staging. Na gałęzi main umieszczałyśmy nowe, w pełni działające wersje naszego programu (np. po wprowadzeniu nowej funkcjonalności), gałąź development służyła jako główna gałąź, na której umieszczane były wszystkie commity, natomiast gałąź Staging służyła nam w końcowej fazie, gdy pracowaliśmy nad już prawie skończoną aplikacją.



Wycinek z historii commitów z githuba: branch Main:

## Staging:

-o-	Commits on Jan 13, 2023
	<b>Wysyłanie formularzy kontaktowych</b> kacperpap committed 4 days ago
	<b>Podstrona HELP</b> kacperpap committed 4 days ago
-o-	Commits on Jan 12, 2023
	<b>Obsługa błędu nieaktywnego konta</b> kacperpap committed 4 days ago
	<b>Link aktywacyjny konta</b> kacperpap committed 4 days ago
	<b>Node-mailer</b> kacperpap committed 4 days ago
	<b>Waluty i akcje z najnowszego dnia</b> Qba02 committed 4 days ago
	<b>Obsługa logowania użytkownika</b> Qba02 committed 5 days ago
	<b>Obsługa wyjątków rejestracji</b> Qba02 committed 5 days ago
	<b>Przebudowa struktury katalogów, w pełni odrębne katalogi backend oraz...</b> ... kacperpap committed 5 days ago
-o-	Commits on Jan 11, 2023
	<b>Działająca rejestracja (narazie bez errorów)</b> Qba02 committed 5 days ago
-o-	Commits on Jan 10, 2023
	<b>Caly frontend oparty na rekcie i wyświetlenie akcji</b> Qba02 committed last week

## Development:

-o- Commits on Dec 27, 2022

Druga wersja formularza logowania

Qba02 committed 3 weeks ago

 b028152 

Pierwsza wersja formularza logowania

Qba02 committed 3 weeks ago

 6e098d9 

Struktura podstron

Qba02 committed 3 weeks ago

 2c61e9d 

-o- Commits on Dec 24, 2022

Zmiana nazwy katalogu scss --> css

kacperpap committed 3 weeks ago

 9eae6df 

Dodanie modelu User do bazy danych

kacperpap committed 3 weeks ago

 26f56cf 

-o- Commits on Dec 23, 2022

Pierwsza wersja strony startowej

Qba02 committed last month

 063d636 

-o- Commits on Dec 21, 2022

Update README.md

awi20221 committed last month

Verified  ab082fd 

Aktualizacja README.md

kacperpap committed last month

 788d810 

Struktura katalogów oraz połączenie do bazy danych MongoDB

kacperpap committed last month

 6f22353 

## 13. Testy jednostkowe

Testy jednostkowe to rodzaj testów, które służą do sprawdzenia poprawności działania poszczególnych elementów (tzw. „jednostek”) w aplikacji. Jednostkami mogą być np. funkcje, metody lub klasy. Testy jednostkowe pozwalają zweryfikować czy dana jednostka działa prawidłowo zgodnie z założeniami i czy zwraca oczekiwane wyniki. Do przeprowadzenia testów jednostkowych naszej aplikacji wykorzystaliśmy popularny framework **Jest** współpracujący ze środowiskiem Node.js. Poniżej testy wykonane dla kluczowego modułu logowania aplikacji, które weryfikują działanie dwóch endpoint’ów serwera:

```
10  describe('register', () => {
11
12    beforeEach(async () => {
13      await mongoose.connect(MONGO_URI, { useNewUrlParser: true, useUnifiedTopology: true });
14      const result = await userModel.findOne({fullName: "testowy", login: "test", email: "test@gmail.com"});
15      if(result !== null){
16        userModel.findOneAndDelete({fullName: "testowy", login: "test", email: "test@gmail.com"})
17      }
18    })
19
20    afterEach(async () => {
21      await userModel.findOneAndDelete({login: 'test', email: 'test@gmail.com'})
22      await mongoose.connection.close();
23    });
24
25    it('should register user', async () => {
26
27      const response = await request('http://localhost:3001/api')
28        .post('/auth/register')
29        .send({fullName: "test", login: "test", email: "test@gmail.com", password: "test", role: 'USER'})
30
31      expect(response.text).toBe('User created successfully, please click the activation link on your mail')
32
33      const checkDB = await userModel.find({email: 'test@gmail.com', login: 'test', role: 'USER'})
34      expect(checkDB[0]).not.toBe(undefined)
35
36    });
37  });
38});
```

*Test jednostkowy funkcji kontrolera rejestracji*

```

41  describe('login', () => {
42
43    beforeEach(async () => {
44      await mongoose.connect(MONGO_URI, { useNewUrlParser: true, useUnifiedTopology: true });
45      const result = await userModel.findOne({fullName: "testowy", login: "test", email: "test@gmail.com"});
46      if(result !== null) {
47        const user = new User({
48          fullName: "testowy",
49          role: "USER",
50          login: "test",
51          email: "test@gmail.com",
52          active: true
53        });
54        const password = "test";
55        await User.register(user, password)
56      }
57    })
58
59    afterEach(async () => {
60      await userModel.findOneAndDelete({login: 'test', email: 'test@gmail.com'})
61      await mongoose.connection.close();
62    });
63
64    it('Should return access token', async () => {
65      const response = await request('http://localhost:3001')
66        .post('/auth/login')
67        .send({login: 'test', password: 'test'})
68
69      if(response.data !== undefined){
70        let authHeader = "Bearer " + response.data;
71        console.log(authHeader);
72        const checkResponse = await request('http://localhost:3001')
73          .get('/users')
74          .set({Authorization: authHeader})
75
76        expect(checkResponse.code).toBe(200)
77        expect(checkResponse.body).not.toBe(undefined)
78      }
79    })
80  })

```

### *Test jednostkowy funkcji kontrolera logowania*

```

PASS | tests/api_auth.test.js
register
  ✓ should register user or return appropriate response (705 ms)
login
  ✓ Should return access token (3 ms)

```

Do testowania reszty ścieżek dostępnych jako API naszego serwera, użyliśmy również narzędzia **Postman**, umożliwiającego wysyłanie zapytań do serwera aplikacji.



Przykład metody http GET wywołanej dla endpoint'a /api/shares/name/ALLEGRO oraz kolekcja testowanych adresów URL po lewej stronie

```

425  },
426  {
427    "slug": "usd-48",
428    "_id": "63b33244ef42f134bc417f5a",
429    "currency": "dolar amerykański",
430    "code": "USD",
431    "mid": 4.4351,
432    "effectiveDate": "2022-12-09",
433    "__v": 0
434  },
435  {
436    "slug": "usd-49",
437    "_id": "63b33245ef42f134bc417fc3",
438    "currency": "dolar amerykański",
439    "code": "USD",
440    "mid": 4.4424,
441    "effectiveDate": "2022-12-12",
442    "__v": 0
443  },
444  {
445    "slug": "usd-50",
446    ...
  
```

Przykład metody http GET wywołanej dla endpoint'a http://localhost:3000/api/currencies/code/USD, który zwraca wszystkie notowania USD znajdujące się w bazie danych

## 14. Przykłady refaktoryzacji kodu

**Refaktoryzacja** istniejącego kodu polega na przebudowaniu istniejącego kodu w celu zwiększenia jego czytelności i zrozumiałości ułatwiającej dalsze utrzymywanie go, zwiększenia wydajności algorytmicznej tj. złożoności obliczeniowej i/lub pamięciowej lub ujednolicenia kodu całego projektu.

### Przykład 1:

Dzięki zauważeniu, że kod css powtarza się dla kilku elementów mogliśmy zastosować jedną z funkcjonalności tego języka i zapisać fragmenty kodu obowiązujące dla wszystkich tych elementów w jednym miejscu dodając ich nazwy po przecinku, dzięki czemu usunęliśmy część zbędnego kodu.

```
85 + .delete-user-header,
86 + .change-password-header,
87 .change-email-header {
88     display: flex;
89     justify-content: center;
90     /*align-items: center;*/
91 -     margin-left: -700px;
92 -     margin-top: 50px;
93 +     margin-top: 30px;
94 }
95 + #delete-user-text,
96 + #change-password,
97 #change-email {
98     position: relative;
99     font-weight: normal;
100    font-size: 28px;
101
102 + #delete-user-text::after,
103 + #change-password::after,
104 #change-email::after {
105     content: "";
```

```
- #input-change-email {
-     width: 25%;
-     color: black;
- }
-
- #button-change-email {
-     margin-left: 50px;
- }
-
- .delete-user-header {
-     display: flex;
-     justify-content: center;
-     margin-left: -800px;
-     margin-top: 50px;
- }
-
- #delete-user-text::after {
-     content: "";
-     position: absolute;
-     left: 0;
-     bottom: -5px;
-     width: 580%;
-     border-bottom: 2px solid;
-     color: hsla(210, 18%, 87%, 1);
- }
```

### Przykład 2:

Zmiana dotycząca przenoszenia użytkownika bezpośrednio na stronę główną po zalogowaniu zamiast wyświetlania komunikatu sprawiła, że wprowadzając jedną linijkę kodu więcej mogliśmy się pozbyć całej sekcji "LogSuccess" oraz całej logiki odpowiedzialnej za wyświetlenie tego komunikatu, co skutkowało tym, że kod stał się czytelniejszy.

```
+          navigate(from, { replace: true });

65      79
66      80      return (
67      81          <div className="container-login">
68          -      {success ? (
69              -          <section className="LogSuccess">
70                  -              <h1 className="LogSuccessText">Zalogowałeś się pomyślnie</h1>
71                  -              <p>
72                      -                  <Link to="/" className="HomeLink">
73                          Home Page
74                      -                  </Link>
75                  -              </p>
76              -          </section>
77          -      ) : (
```

### Przykład 3:

```
1 usage  ▲ Kacper
13     □ async function verifyRequestAvailability(req, res, next){
14         □     if( String(req.user.role) === "ADMIN"){
15             □         return true;
16         } else {
17             □                 const userToModify = await User.find( filter: {'slug': req.params.slug});
18             □                 if (String(req.user._id) === String(userToModify[0]._id)) {
19                 □                     return true;
20             }
21             □         return false;
22         }
23     }

1 usage  ▲ Kacper
25     □ async function verifyIfAdmin(req, res, next){
26         □     if( String(req.user.role) === "ADMIN"){
27             □         return true;
28         } else
29             □         return false;
30     }
```

Powyższe funkcje posłużyły jako weryfikacja czy użytkownik próbujący wykonać daną akcję np. zmianę hasła jest rzeczywiście użytkownikiem którego dane chce zmienić, bądź administratorem posiadającym takie uprawnienia. Wprowadzenie tych 2 funkcji pozwoliło na wyeliminowanie każdorazowego sprawdzania „tożsamości użytkownika” przy poszczególnych zapytaniach.

```
95  ↗async function remove(req, res, next) {  
96    ↗  if(await authController.verifyRequestAvailability(req,res,next)) {  
97      const user = await User.findOne( filter: { 'slug': req.params.slug});  
98      if (!user) return next();  
99      await user.remove();  
100     return res.status(200).send({message: 'User was removed'});  
101   }  
102   ↗return res.status(401).send("You do not have access to data you are trying to modify. Please log in to accurate account");  
103 }
```

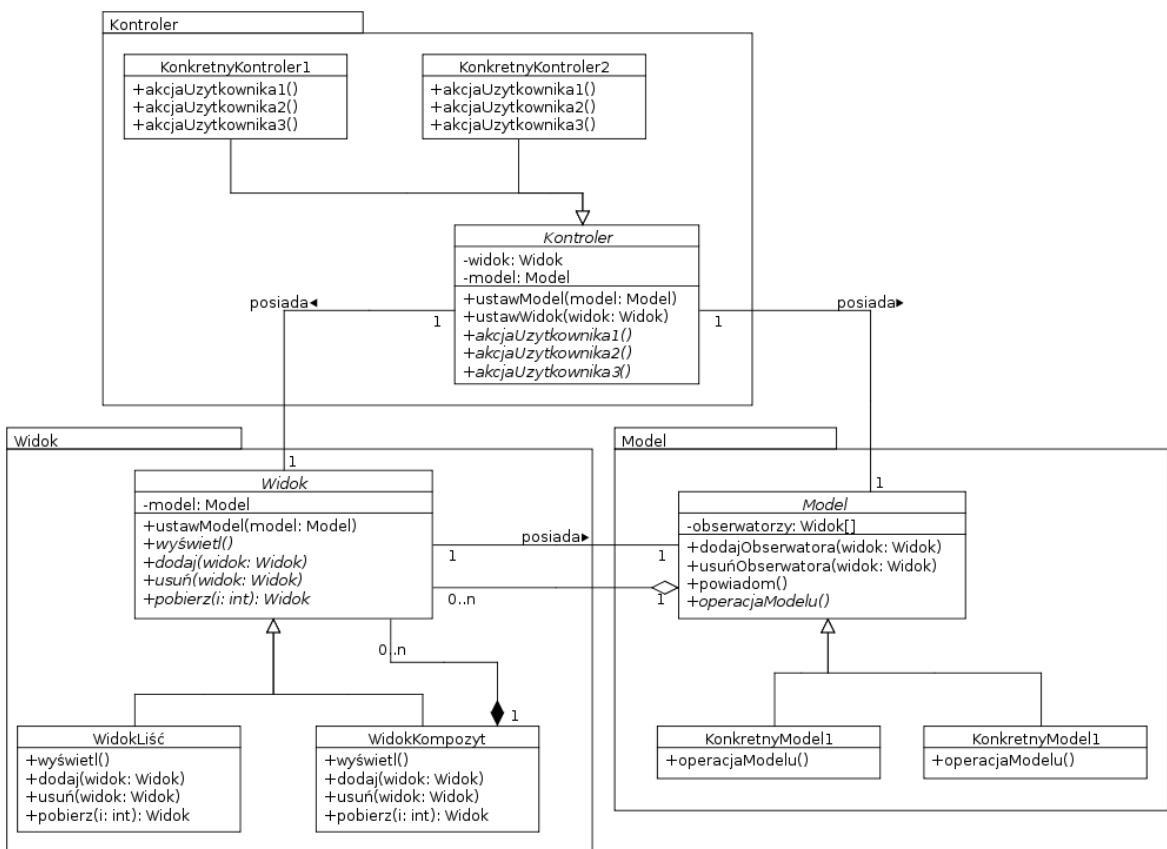
*Nowa wersja funkcji*

```
106  ↗async function remove(req, res, next) {  
107    ↗  if (!req.user || (req.user.role !== "ADMIN" && String(req.user._id) !== String(req.params.slug))) {  
108      ↗    return res.status(401).send({ message: "Unauthorized" });  
109    }  
110  
111    const user = await User.findOne( filter: { 'slug': req.params.slug});  
112    ↗  if (!user) {  
113      ↗    return res.status(404).send({ message: "User not found" });  
114    }  
115  
116    ↗  try {  
117      ↗    await user.remove();  
118      ↗    return res.status(200).send({ message: 'User was removed' });  
119    } catch (error) {  
120      ↗    return res.status(500).send({ message: "Error deleting user" });  
121    }  
122  }  
123 }
```

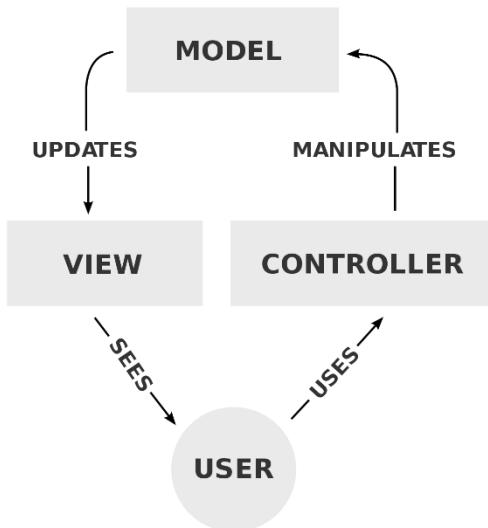
*Stara wersja funkcji*

## 15. Przykład wzorca projektowego

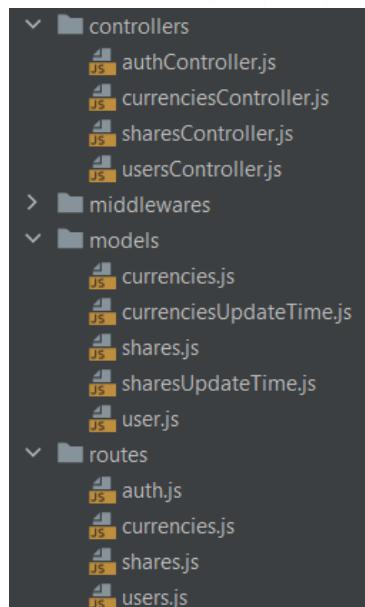
MVC (model-view-controller) to wzorzec służący do organizowania struktury aplikacji, które posiadają graficzne interfejsy użytkownika. Został on zaprojektowany w 1979 roku przez pracownika firmy Xerox. Zakłada on podział aplikacji na trzy odseparowane od siebie części: model – stanowiący logikę aplikacji, widok- opis wyświetlanego modelu w ramach interfejsu użytkownika oraz kontroler – przejmujący dane wejściowe i odpowiednio modyfikujący model.



MVC jest bardzo popularny pośród aplikacji webowych. Implementacja wzorca jest jednak zmodyfikowana ze względu na brak aktywnych modeli, co wynika z działania protokołu HTTP – serwer nie ma możliwości wysyłania odświeżonego widoku bez pojawiения się żądania od użytkownika. Stosowanie wzorca pozwala na separację poszczególnych części systemu dzięki czemu ułatwia modyfikacje często zmieniających się interfejsów użytkownika oraz warstwy prezentacji bez zmiany logiki aplikacji. Ponadto, dzięki braku zależności pomiędzy modelem oraz widokami, możliwe jest współistnieje wielu widoków prezentujących te same dane na różne sposoby. Główną wadą wzorca jest wprowadzenie dużej złożoności aplikacji. Ogólny zarys kooperacji części przedstawiony jest na poniższym diagramie:



Wzorzec MVC został przez nas częściowo zastosowany w uproszczonej formie, po stronie serwera aplikacji w celu utworzenia API. Nie jest to dokładne zastosowanie wzorca, ponieważ nie dzieli całości aplikacji tylko stronę serwerową, oraz upraszcza widoki do postaci plików *plain/text* lub *application/json*. Modelem są kolekcje tworzone w bazie danych, które przechowują wszystkie dane trwałe wykorzystywane w aplikacji (pliki w katalogu *models*). Jest on odseparowany od kontrolera, który zajmuje się obsługiwaniem żądań HTTP, odpowiednio modyfikując dane w bazie (katalog *controllers*). Widokiem jest nasze API, które stanowi interfejs dla części frontend'owej naszej aplikacji, udostępniając jej odpowiednio zmodyfikowane przez kontroler dane udostępniane jako HTTP response dzięki framework'owi Express.

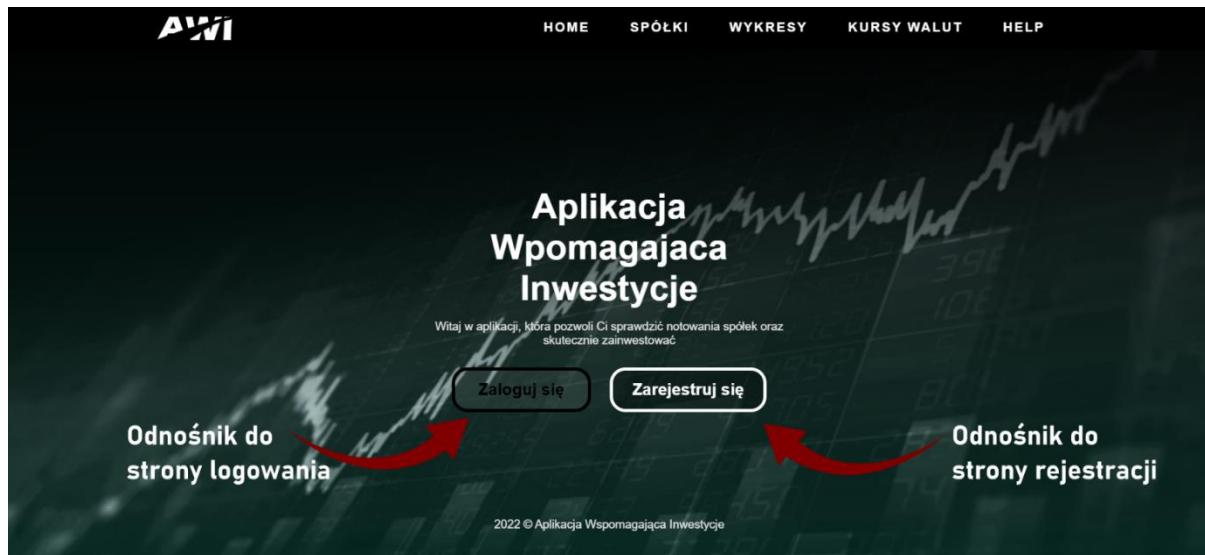


*Struktura katalogów odpowiadająca wzorcowi MVC, odseparowanie modelów od kontrolerów oraz ścieżki API stanowiące widoki serwera, czyli pliki w formacie plain/text oraz application/json*

## 16. Instrukcja użytkowania programu

### 1. Instrukcja ogólna:

Strona powitalna/startowa, na którą trafia użytkownik po uruchomieniu aplikacji:



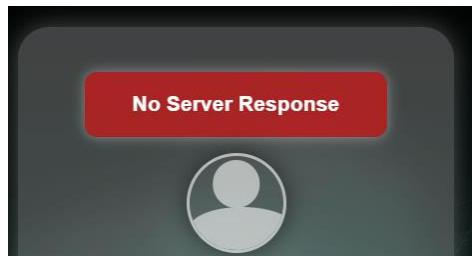
Po wybraniu jednego z przycisków przeniesienie do jednego z formularzy, które wyglądają następująco:

### Strona logowania

### Strona rejestracji

The image displays two side-by-side forms. The left form is labeled "Formularz logowania" (Login form) and contains fields for "Login" and "Hasło" (Password), with a "Zaloguj się" (Log in) button at the bottom. The right form is labeled "Formularz rejestracji" (Registration form) and contains fields for "Imię" (Name), "Login", "Podaj email" (Provide email), and "Wprowadź hasło" (Enter password), with a "Zarejestruj się" (Register) button at the bottom. Both forms include a placeholder text "Nie masz konta? Zarejestruj się" (Don't have an account? Register) and a "Odnośnik do strony rejestracji" (Link to the registration page) with a red arrow pointing to it. The footer of both forms includes the text "2022 © Aplikacja Wspomagająca Inwestycje".

W przypadku błędów pojawią się pomocne komunikaty wyglądające następująco:



Po udanej rejestracji komunikat o sprawdzeniu skrzynki odbiorczej, po kliknięciu w link aktywacyjny następuje przekierowanie na stronę logowania.



Natomiast po zalogowaniu wyświetla się strona główna z opcjami wyboru

## Strona główna

The screenshot shows the main page of the AWI application. At the top center is the AWI logo. Below it, the text "Co chcesz zrobić?" is displayed, followed by the instruction "Wybierz jedną z opcji". There are four rectangular buttons arranged in a 2x2 grid:

- Zobacz aktualne notowania spółek
- Zobacz aktualne kursy walut
- Przejdź do strony z wykresami
- Potrzebujesz pomocy?

Red arrows point to several elements:

- A red arrow points from the bottom left towards the "Stopka" (Footer) section.
- A red arrow points from the right side towards the text "Odnośniki do konkretnych stron".
- A red box highlights the entire 2x2 grid of buttons.
- A red box highlights the "Stopka" section at the bottom.
- A red box highlights the "Wyloguj" (Logout) button at the bottom right.

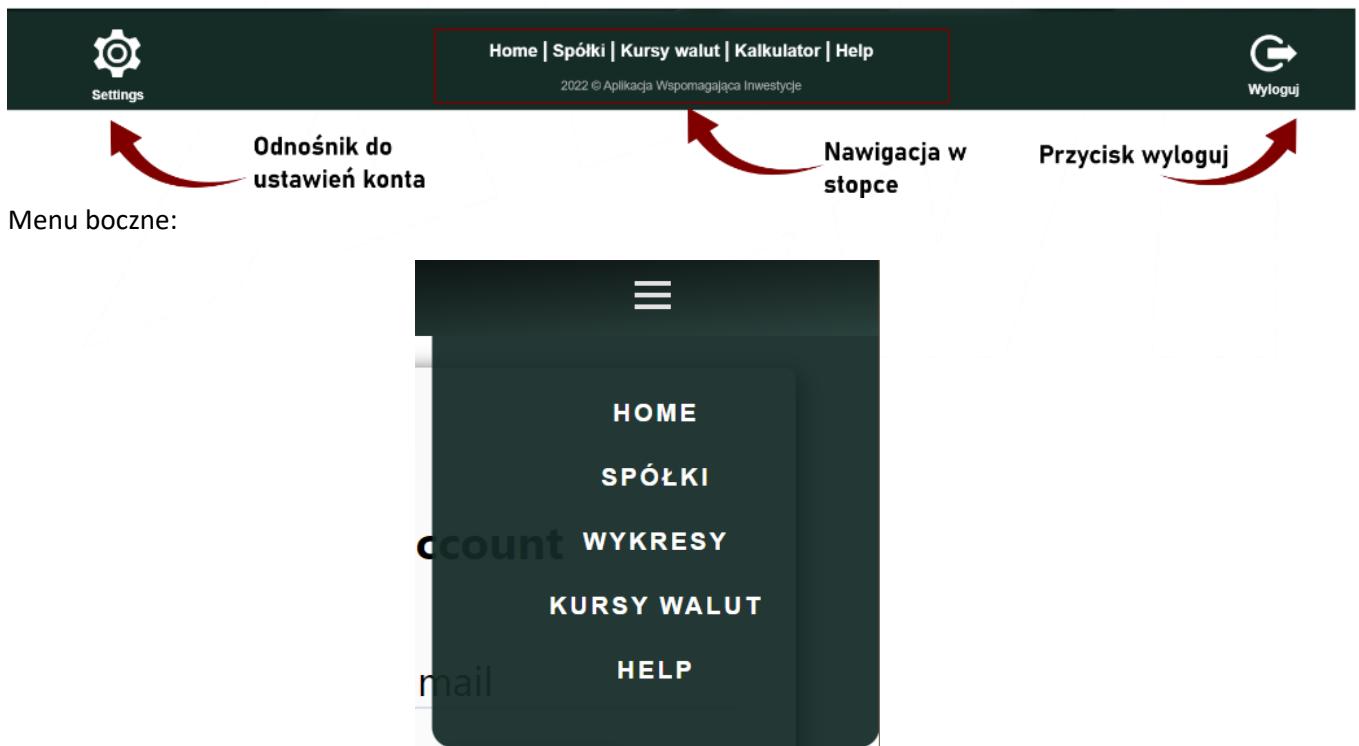
At the very bottom, there is a footer bar with icons for "Settings" (gear), "Home", "Spółki", "Kursy walut", "Kalkulator", "Help", and "Wyloguj".

W większości z podstron znajdują się następujące elementy, które służą nawigowaniu po stronie. W stopce dodatkowo znajduje się przycisk do wylogowania oraz odnośnik do strony umożliwiającej wprowadzenie zmian w koncie. Dodatkowo logo po kliknięciu zazwyczaj przekierowuje do strony głównej.

## Nawigacja



## Stopka



## 2. Wskazówki dla poszczególnych podstron:

### Strona wykresów

The screenshot shows a dark-themed web application interface. At the top left is the logo 'AWI'. Below it is a form titled 'Typ' with dropdown menus for 'Instrument pierwszy' (set to 'waluty') and 'Instrument drugi' (set to 'wybierz'). A large dropdown menu lists various currencies: AUD, BRL, BGN, CNY, DKK, EUR, PHP, HKD, IDR, ISK, JPY, CAD, KRW, MYR, MXN, NOK, NZD, ZAR, RON. A red arrow points from the right towards this dropdown menu. To the right of the form, there is descriptive text in Polish: 'Formularz z opcjami wyboru 2 spółek/walut i porównania ich wykresów'.

### Strona spółek/walut

The screenshot shows a table titled 'Kursy walut' with the subtitle 'Wyszukaj: 416 records...'. Above the table are two input fields: 'Najmłodsza data: 2023-01-11' and 'YYYY-MM-DD', followed by a button 'Wyszukaj datę'. A red arrow points from the left towards these filtering options with the text 'Piltrowanie zawartości tabeli'. Another red arrow points from the right towards the 'Wyszukaj datę' button with the text 'Pobierz z bazy dane z konktnego dnia'. To the right of the table, there is descriptive text in Polish: 'Tabela z opcjami sortowania danych'.

name	minimalRate	maximalRate	change	effectiveDate
3RGAMES	0.436	0.465	0.45	2023-01-11
08OCTAVA	1.02	1.02	0	2023-01-11
AGROTON	3.48	3.55	-2.35	2023-01-11
ACTION	14.44	14.7	0	2023-01-11
ADIUVO	0.85	0.89	0	2023-01-11
11BIT	592	620	-2.92	2023-01-11
06MAGNA	3.755	3.88	-1.81	2023-01-11

# Strona Help

The screenshot shows a contact form with three input fields: 'Name', 'Email', and 'Message'. Below the message field is a 'Send Message' button. At the bottom of the form, there is a note: 'Or send directly to: awi2022.1.0@gmail.com'. A red arrow points from the text 'Formularz kontaktowy, który wyśle wiadomość na maila awi2022.1.0@gmail.com' to the 'Send Message' button.

CONTACT US

Name:  
Podaj imię

Email:  
Podaj e-mail

Message:  
Miejsce na twoją wiadomość...

Send Message

Or send directly to: awi2022.1.0@gmail.com

Formularz kontaktowy, który wyśle wiadomość na maila awi2022.1.0@gmail.com

# Strona zarządzania kontem

The screenshot shows a personal account page with a user icon and the text 'Your personal account' and 'User: "jakub"'. It features three main buttons: 'Zmień adres e-mail', 'Zmień hasło', and 'Usuń konto'. A red arrow points from the text 'Zmień hasło' to the 'Zmień hasło' button. Another red arrow points from the text 'Usuń swoje konto (nieodwracalna operacja)' to the 'Usuń konto' button. A third red arrow points from the text 'Zmień email' to the 'Zmień e-mail' button.

Your personal account  
User: "jakub"

Zmień adres e-mail

Wpisz nowy e-mail

Zmień e-mail

Zmień hasło

Wpisz nowe hasło

Zmień hasło

Usuń konto

Usuń konto

Zmień hasło

Zmień email

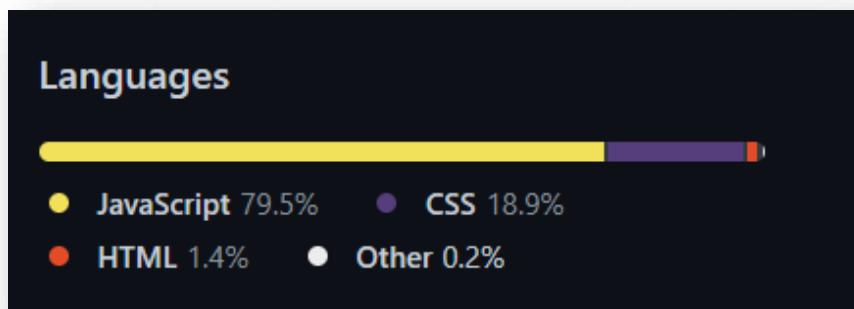
Usuń swoje konto  
(nieodwracalna operacja)

## 17.Podsumowanie

Opis prac, problemy napotkane podczas realizacji projektu, cechy oraz funkcjonalności wyróżniające opracowane rozwiązanie)

Głównym założeniem naszego projektu “Aplikacja Wspomagająca Inwestycje” było stworzenie platformy, która zgromadzi w jednym miejscu jak najwięcej informacji biznesowych/giełdowych/inwestycyjnych. To założenie udało nam się zrealizować, niemniej jednak napotkaliśmy po drodze kilka trudności natury technicznej - dużym problemem okazała się trudność w znalezieniu darmowych i zarazem aktualnych danych giełdowych, problem ten udało nam się rozwiązać korzystając z archiwalnych tabel udostępniane przez NPB. Kolejnym problemem okazała się złożoność pracy nad projektem w grupie (nawet tak niewielkiej), a wnioskiem jaki wyciągamy na przyszłość jest to, że im częstsza komunikacja między członkami zespołu tym lepiej.

Korzystaliśmy z szeregu różnych technologii webowych: node.js, React, express, mongoDB, jwt, passport, git oraz pisaliśmy w językach JavaScript, CSS oraz HTML. Przez to nasz projekt rozbudował się do sporych rozmiarów, plusem jest natomiast jego potencjał do dalszej rozbudowy.



Nasza aplikacja posiada szereg funkcjonalności takich jak:

- Możliwość utworzenia konta, zalogowania się oraz edycji wprowadzonych danych
- Przedstawianie w przystępny sposób wspomnianych wcześniej danych, opcja ich filtrowania oraz sortowania
- Możliwość porównania ze sobą wykresów dwóch spółek/kursów walut

W planie mieliśmy również wprowadzić kolejnych opcji takich jak:

- Zakładka “kalkulatora zysków” do symulacji i przewidywania
- Opcja zapisywania preferencji wyboru użytkownika na jego koncie
- Kalkulator ryzyka szacujący bezpieczeństwo inwestycji ,itp.

**Podział zadań:**

Wiktor Paleczny	Jakub Nowak	Kacper Papuga
<b>Główna odpowiedzialność - projekt</b>		
Zakładka porównywania wykresów	Frontend aplikacji – stworzenie GUI całej aplikacji, pobranie i wyświetlanie aktualnych danych, szablony i style podstron, algorytm sortowania i filtrowania danych, obsługa logowania i rejestracji użytkowników oraz stworzenie kontekstu	Backend aplikacji - połączenie z bazą danych, stworzenie API, kontrolerów, middleware'ów, modeli, ścieżek, pobieranie aktualnych danych do bazy, podstrona pomocy i ustawień, obsługa logowania i rejestracji użytkowników oraz kontrola przepływu danych między komponentami
<b>Dokumentacja</b>		
Pomoc przy tworzeniu diagramów	Słownik pojęć i terminów	Specyfikacje projektu
Wymagania niefunkcjonalne	Diagramy sekwencji	Opis wybranej metodyki wytwarzania
	Diagram komponentów	Diagram przypadku użycia
	Diagram wdrożenia	Wymagania funkcjonalne
	Przykłady współpracy zespołu	Diagramy wymagań
	Przykład refaktoryzacji kodu	Wykaz zastosowanych technologii
	Instrukcja użytkowania programu	Diagramy aktywności
	Podsumowanie	Testy jednostkowe
	Diagram klas	Przykład wzorca projektowego,
		Spis treści i strona tytułowa
		Diagram klas
		Diagramy sekwencji