

AKADEMIA NAUK STOSOWANYCH
W NOWYM SĄCZU

WYDZIAŁ INŻYNIERYJNY

Informatyka Stosowana

DOKUMENTACJA PROJEKTOWA
INŻYNIERIA OPROGRAMOWANIA

Botex

Autorzy:
Szymon Zwoliński
Piotr Szczepanek

Prowadzący:
mgr inż. Daniel Drozd

Nowy Sącz 2022

Spis treści

1. Tytuł roboczy projektu	3
2. Cel projektu	4
3. Grupa docelowa	5
4. Nasz bot, a konkurencja	6
5. Wymagania funkcjonalne	7
6. Wymagania niefunkcjonalne	8
7. Możliwe problemy w procesie implementacji	9
8. Opis technologii	10
9. Diagram przypadków użycia	11
10. Diagram ERD	16
11. Diagram klas	17
12. Diagram aktywności	18
13. Diagram sekwencji	19
14. Rozpoznanie wzorców projektowych	20
15. Link do repozytorium	27
16. Wnioski	28
17. Bibliografia	29
Literatura	30
Spis rysункów	30

1. Tytuł roboczy projektu

Tytuł roboczy Naszego projektu to - Botex.

2. Cel projektu

Celem Naszego projektu było stworzenie bota, mającego możliwość interakcji z Użytkownikiem. Bot, w zależności od potrzeb, będzie posiadał różnorakie funkcje takie jak: publikowanie treści w serwisie Twitter, zbieranie informacji o Użytkownikach serwisu Instagram, wyszukiwanie zagadnień z Wikipedii, wyszukiwanie informacji w wyszukiwarce Google, wysyłanie wiadomości e-mail.

Nasza strona korzysta z bazy danych, w której zawarta jest baza użytkowników, domyślnych wiadomości - są one dodawane przez Administratora, monitoring akcji uruchamianych przez Użytkownika z wiadomościami.

3. Grupa docelowa

Grupą docelową Naszego bota są osoby, które chcą usprawnić sobie korzystanie z social mediów, bez zbędnego otwierania dużej ilości kart. Każdy znajdzie tutaj coś dla siebie. Zarówno influencerzy, jak i inne osoby, które na codzień korzystają z serwisów społecznościowych.

4. Nasz bot, a konkurencja

Na rynku jest wiele oprogramowań tego typu. Brightery Twitter Bot, ale też masa innych. Oczywiście, są one darmowe, ale w zamian za to zasypują Użytkownika masą reklam oraz jak można się domyślać, potrafią gromadzić dużo danych na Jego temat.

Nasz bot taki nie będzie. Będzie on zoptymalizowany, a także nie planujemy wdrażania do niego reklam. Dlatego mamy nadzieję, że przede wszystkim poprzez Nasze poszanowanie prywatności Użytkowników, a także brak reklam, będzie on się znacząco z biegiem czasu wybijał ponad konkurencję.

5. Wymagania funkcjonalne

Wymagania funkcjonalne dla Naszego bota to:

- pobranie od Użytkownika danych autentykacyjnych podczas wysyłania wiadomości e-mail, publikowania Twittów, w serwisie Instagram. **MVP**

Funkcje, z których może skorzystać Użytkownik, to:

- publikowanie Tweetów **MVP**
- zbieranie danych z serwisu Instagram
- możliwość wysyłania wiadomości, znalezionych w Wikipedii, czy też w wyszukiwarce Google **MVP**
- wyszukiwanie treści podawanych przez Użytkownika **MVP**

6. Wymagania niefunkcjonalne

System, odpowiedzialny za wszystkie powyższe opcje, opracowany jest w oparciu o framework .Net. Do stworzenia Naszego bota, skorzystaliśmy z języka C. Przeznaczeniem systemu miało być jak najszerze grono odbiorców. Do testowania bota skorzystamy z oprogramowania Selenium. Połączenie z bazą danych będzie zrealizowane poprzez Nuget Entity Framework

Nasza strona jest zoptymalizowana pod kątem prędkości jej działania.

W celu zaimplementowania bazy danych, skorzystamy z SQLite.

7. Możliwe problemy w procesie implementacji

Podczas implementacji mogą oczywiście wystąpić niechciane problemy, których nie będziemy się spodziewać. Jednak na tą chwilę ciężko Nam stwierdzić czego mogą one dotyczyć. Mamy nadzieję, że plan jaki mamy na Naszego bota, przebiegnie bez jakiś większych problemów.

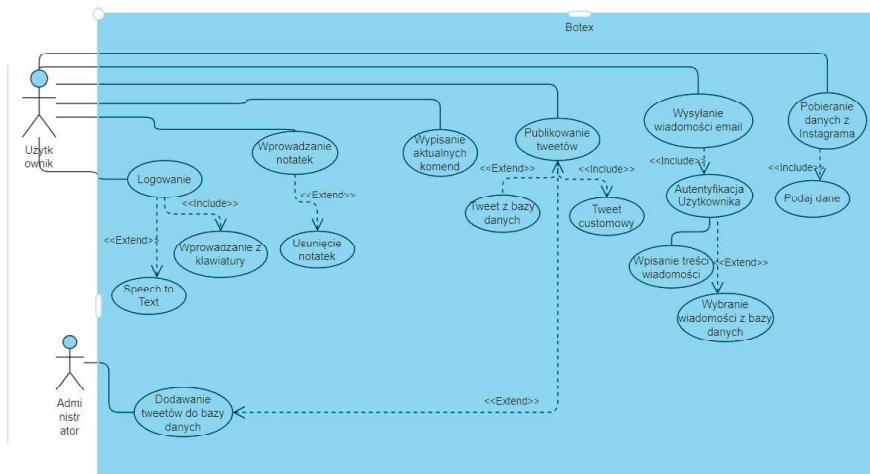
8. Opis technologii

.NET Framework, w skrócie .NET (wym. dot net) – platforma programistyczna opracowana przez Microsoft, obejmująca środowisko uruchomieniowe (Common Language Runtime – CLR) oraz biblioteki klas dostarczające standardowej funkcjonalności dla aplikacji. Technologia ta nie jest związana z żadnym konkretnym językiem programowania, a programy mogą być pisane w jednym z wielu języków – na przykład C++/CLI, C, F, J, Delphi 8 dla .NET, Visual Basic .NET. Zadaniem platformy .NET Framework jest zarządzanie różnymi elementami systemu: kodem aplikacji, pamięcią i zabezpieczeniami.

Selenium – zautomatyzowana platforma testowa dla aplikacji internetowych[1]. Selenium dostarcza narzędzie odtwarzania zadań w celu tworzenia powtarzalnych testów funkcjonalnych bez konieczności uczenia się testowego języka skryptowego (Selenium IDE). Częścią frameworka jest również dziedzinowy język testowy (Seleneese), który służy do pisania testów w różnych językach programowania, w tym JavaScript (Node.js), C, Groovy, Java, Perl, PHP, Python, Ruby i Scala. Testy można następnie uruchomić na większości nowoczesnych przeglądarek internetowych. Selenium działa w systemach Windows, Linux i macOS. Jest to oprogramowanie typu open source wydane na licencji Apache License 2.0.

SQLite – otwartoźródłowy system zarządzania relacyjną bazą danych oraz biblioteka C implementująca taki system, obsługująca SQL. Została stworzona przez Richarda Hippa[1] i jest dostępna na licencji domeny publicznej. Projekt został rozpoczęty w roku 2000. Biblioteka implementuje silnik SQL, dając możliwość używania bazy danych bez konieczności uruchamiania osobnego procesu RDBMS. W wielu zastosowaniach, a w szczególności w systemach wbudowanych, takie rozwiązanie jest najpraktyczniejsze. SQLite posiada również API do innych niż C języków programowania, a mianowicie: ActionScript, Perl, PHP, Ruby, C++, Delphi, Python, Java, Tcl, Visual Basic, platformy .NET i wielu innych; a także interfejs powłokowy. Baza może łączyć się przez ODBC.

9. Diagram przypadków użycia



Rys. 9.1. Diagram przypadków użycia

Przypadek użycia 01:	Logowanie do bota
Aktorzy	Użytkownik
Warunki wstępne	Uruchomiona aplikacja, bot prosi o dane autoryzacyjne
Rezultat	Zalogowanie Użytkownika
Scenariusz główny	<ol style="list-style-type: none"> Wybór opcji pisania z klawiatury. Wybranie opcji Speech to Text. Zatwierdzenie danych autoryzacyjnych.
Scenariusz alternatywny	1.1 Odmowa dostępu, po 3 błędach zamknięcie aplikacji.

Przypadek użycia 02:	Wprowadzenie notatek
Aktorzy	Użytkownik
Warunki wstępne	Zalogowany Użytkownik
Rezultat	Poprawnie wprowadzono notatkę do bazy danych.
Scenariusz główny	<ol style="list-style-type: none"> 1. Wybór polecenia "Stwórz notatkę". 2. Wprowadzenie treści notatki. 3. Zatwierdzenie zapisu.
Scenariusz alternatywny	<ol style="list-style-type: none"> 1.1 Odmowa dostępu. 2.1 Wykroczenie poza limit znaków. 3.1 Odrzucenie zapisu notatki.

Przypadek użycia 03:	Usunięcie notatek
Aktorzy	Użytkownik
Warunki wstępne	Zalogowany Użytkownik.
Rezultat	Usunięcie notatki z bazy danych.
Scenariusz główny	<ol style="list-style-type: none"> 1. Wybór opcji usunięcia notatki. 2. Wybór notatki do usunięcia. 3. Zatwierdzenie usunięcia notatki.
Scenariusz alternatywny	<ol style="list-style-type: none"> 1.1 Odmowa dostępu. 2.1 Brak dostępnych notatek. 3.1 Anulowanie chęci usunięcia notatki.

Przypadek użycia 04:	Wykorzystanie opcji Speech to Text.
Aktorzy	Użytkownik
Warunki wstępne	Uruchomiona aplikacja, zainstalowana paczka z językiem polskim do rozpoznawania głosu, mikrofon.
Rezultat	Poprawnie wprowadzona wiadomość głosowa.
Scenariusz główny	<ol style="list-style-type: none"> 1. Wybór opcji "Speech to Text". 2. Przekazanie treści wiadomości w sposób słowny, z wykorzystaniem mikrofonu. 3. Zatwierdzenie wprowadzenia danych.
Scenariusz alternatywny	<ol style="list-style-type: none"> 2.1 Przekazanie wiadomości w sposób niezrozumiałą dla systemu. 3.1 Możliwość poprawny danych ręcznie.

Przypadek użycia 05:	Publikowanie Tweetów
Aktorzy	Użytkownik
Warunki wstępne	Użytkownik zalogowany.
Rezultat	Opublikowanie Tweeta
Scenariusz główny	<ol style="list-style-type: none"> 1. Wybór opcji publikacji Tweeta. 2. Wybranie opcji publikacji Tweeta poprzez metodę "Speech to Text". 3. Serwis Tweeter nie odrzucił danych logowania bota. 4. Wybór treści Tweeta - z bazy danych lub własna.
Scenariusz alternatywny	<ol style="list-style-type: none"> 2.1 Niezrozumiałły tekst dla Bota. 3.1 Serwis Tweeter odrzucił danych logowania bota. 3.2 Wycofanie do głównego menu bota z informacją o błędzie. <ol style="list-style-type: none"> 4.1 Brak Tweeta w bazie danych. 4.2 Zbyt długa treść Tweeta.

Przypadek użycia 06:	Zapisanie Tweeta do bazy danych
Aktorzy	Administrator
Warunki wstępne	Użytkownik zalogowany.
Rezultat	Dodanie Tweeta do bazy.
Scenariusz główny	<ol style="list-style-type: none"> 1. Wybór opcji dodania Tweeta do bazy danych. 2. Podanie treści Tweeta. 3. Zatwierdzenie treści Tweeta.
Scenariusz alternatywny	<ol style="list-style-type: none"> 1.1 Odmowa dostępu z powodu braku uprawnień. <ol style="list-style-type: none"> 2.1 Zbyt długa treść Tweeta. 3.1 Odrzucenie treści Tweeta.

Przypadek użycia 07:	Zalogowanie się do poczty e-mail
Aktorzy	Użytkownik
Warunki wstępne	Użytkownik zalogowany do Bota.
Rezultat	Zalogowanie użytkownika do poczty e-mail.
Scenariusz główny	<ol style="list-style-type: none"> 1. Wybór opcji wysyłania wiadomości mail. 2. Wybór jednej wspieranej przez bota poczty. 3. Wprowadzenie danych autoryzacyjnych do konta pocztowego.
Scenariusz alternatywny	<ol style="list-style-type: none"> 1.1 Odmowa dostępu z powodu braku uprawnień. <ol style="list-style-type: none"> 2.1 Brak wspieranych poczt. 2.2 Odrzucenie połączenia z pocztą. 3.1 Podanie złych danych autoryzacyjnych. 3.2 Zablokowane konto w usłudze pocztowej.

Przypadek użycia 08:	Wysłanie treści wiadomości
Aktorzy	Użytkownik
Warunki wstępne	Użytkownik zalogowany, Użytkownik zalogowany do usługi poczty.
Rezultat	Wysłanie własnej wiadomości e-mail.
Scenariusz główny	<ol style="list-style-type: none"> 1. Wybór opcji wysłania treści wiadomości. 2. Wpisanie własnej treści wiadomości. 3. Zatwierdzenie treści wiadomości. 4. Wysłanie wiadomości e-mail.
Scenariusz alternatywny	<ol style="list-style-type: none"> 1.1 Odmowa dostępu z powodu braku uprawnień. <ol style="list-style-type: none"> 2.1 Zbyt długa treść wiadomości. 3.1 Odrzucenie treści wiadomości. 4.1 Błąd wysłania wiadomości. 4.2 Błąd połączenia z pocztą.

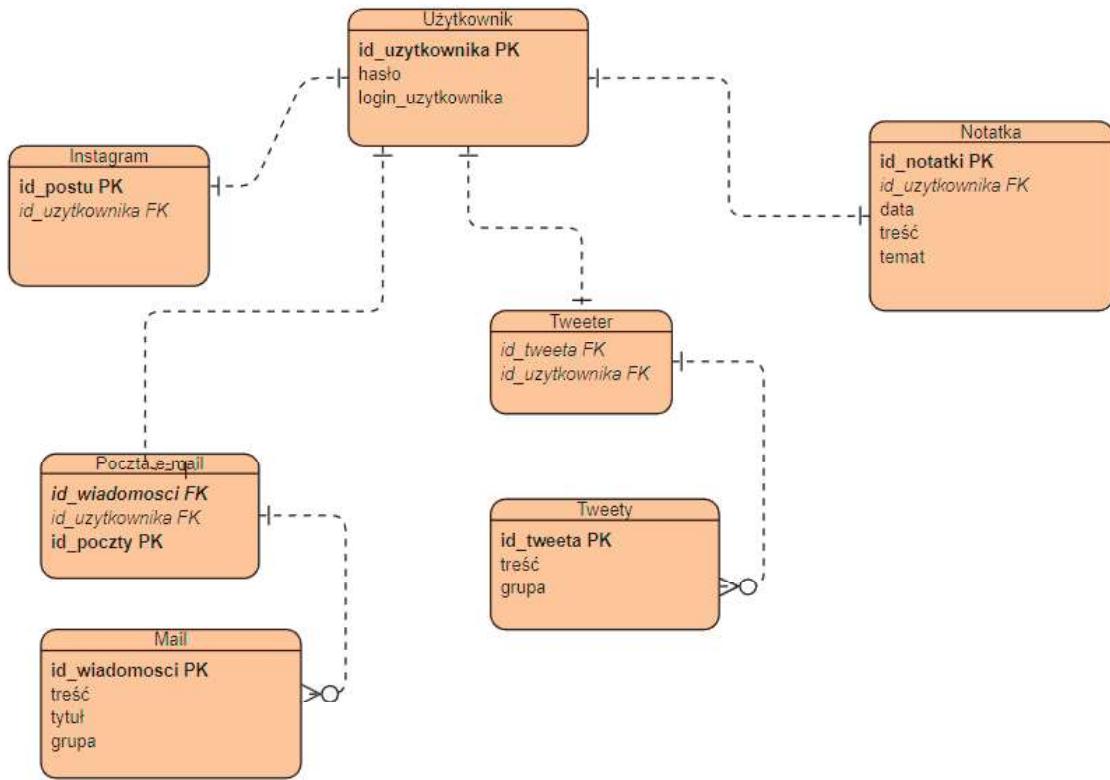
Przypadek użycia 09:	Wysłanie wiadomości z bazy danych
Aktorzy	Użytkownik
Warunki wstępne	Użytkownik zalogowany, użytkownik zalogowany do usługi poczty.
Rezultat	Wysłanie wiadomości e-mail z bazy danych.
Scenariusz główny	<ol style="list-style-type: none"> 1. Wybór opcji dodania wiadomości z bazy danych. 2. Wybór wiadomości z bazy danych. 3. Zatwierdzenie wyboru wiadomości. 4. Wysłanie wiadomości.
Scenariusz alternatywny	<ol style="list-style-type: none"> 1.1 Odmowa dostępu z powodu braku uprawnień. 2.1 Brak wiadomości w bazie danych. 3.1 Odrzucenie treści wiadomości. 4.1 Błąd wysyłania wiadomości. 4.2 Błąd połączenia z pocztą.

Przypadek użycia 10:	Korzystanie z Instagrama
Aktorzy	Użytkownik
Warunki wstępne	Użytkownik zalogowany.
Rezultat	Bot przestawi się w tryb zarządzania Instagramem.
Scenariusz główny	<ol style="list-style-type: none"> 1. Wybór opcji trybu Instagrama. 2. Podanie danych autoryzacyjnych do serwisu Instagram. 3. Zalogowanie się.
Scenariusz alternatywny	<ol style="list-style-type: none"> 1.1 Odmowa dostępu z powodu braku uprawnień. 2.1 Niepoprawne dane autoryzacyjne dla serwisu Instagram. 3.1 Odrzucenie połączenia. 3.2 Zablokowane konto.

Przypadek użycia 11:	Sprawdzenie listy "followers" w serwisie Instagram.
Aktorzy	Użytkownik
Warunki wstępne	Użytkownik zalogowany, użytkownik zalogowany do serwisu Instagram.
Rezultat	Bot wróci liste "follow" z serwisu Instagram.
Scenariusz główny	<ol style="list-style-type: none"> 1. Wybór opcji trybu sprawdzenia followow. 2. Wyświetlenie listy followow danego konta <p style="text-align: right;">heightScenariusz alternatywny</p>
1.1 Odmowa dostępu z powodu braku uprawnień.	<ol style="list-style-type: none"> 2.1 Brak uprawnień od strony serwisu. 2.2 Odrzucenie połączenia.

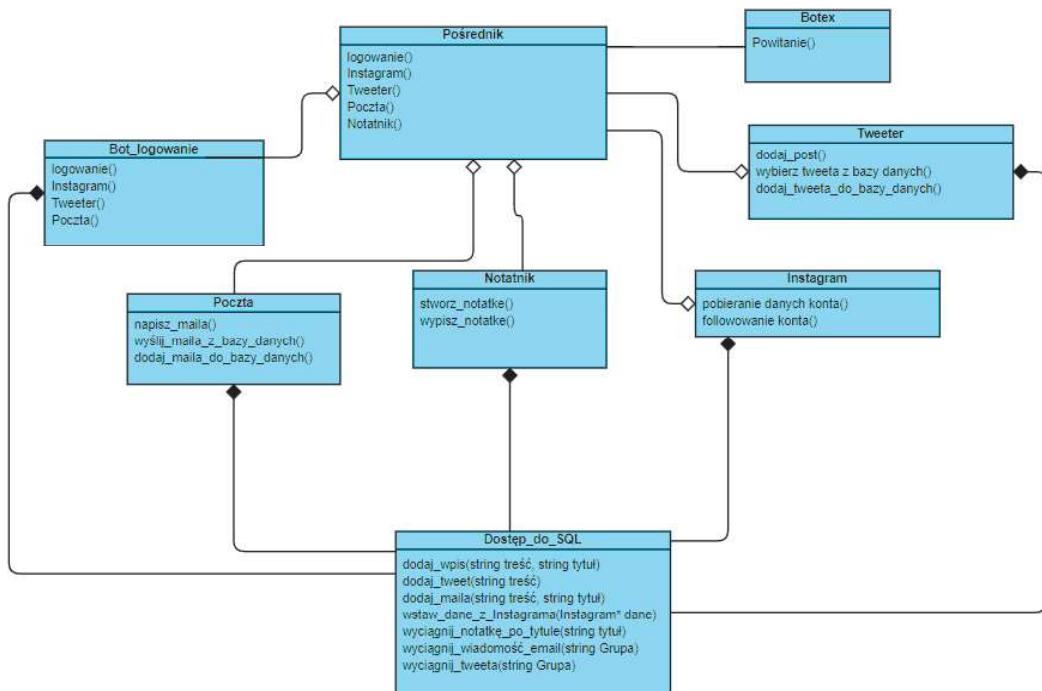
Przypadek użycia 12:	Followowanie konta w Instagramie.
Aktorzy	Użytkownik
Warunki wstępne	Użytkownik zalogowany, użytkownik zalogowany do serwisu Instagram.
Rezultat	Konto do którego jest zalogowany bot zafollowuje podanego użytkownika.
Scenariusz główny	<ol style="list-style-type: none">1. Podanie nazwy użytkownika do follow.2. Zatwierdzenie.
Scenariusz alternatywny	<ol style="list-style-type: none">1.1 Odmowa dostępu z powodu braku uprawnień.1.2 Niepoprawna nazwa użytkownika.2.1 Konto już followuje podanego użytkownika.2.2 Konto nie może followować danego użytkownika.2.3 Odrzucenie połączenia.2.4 Brak uprawnień od strony serwisu.

10. Diagram ERD



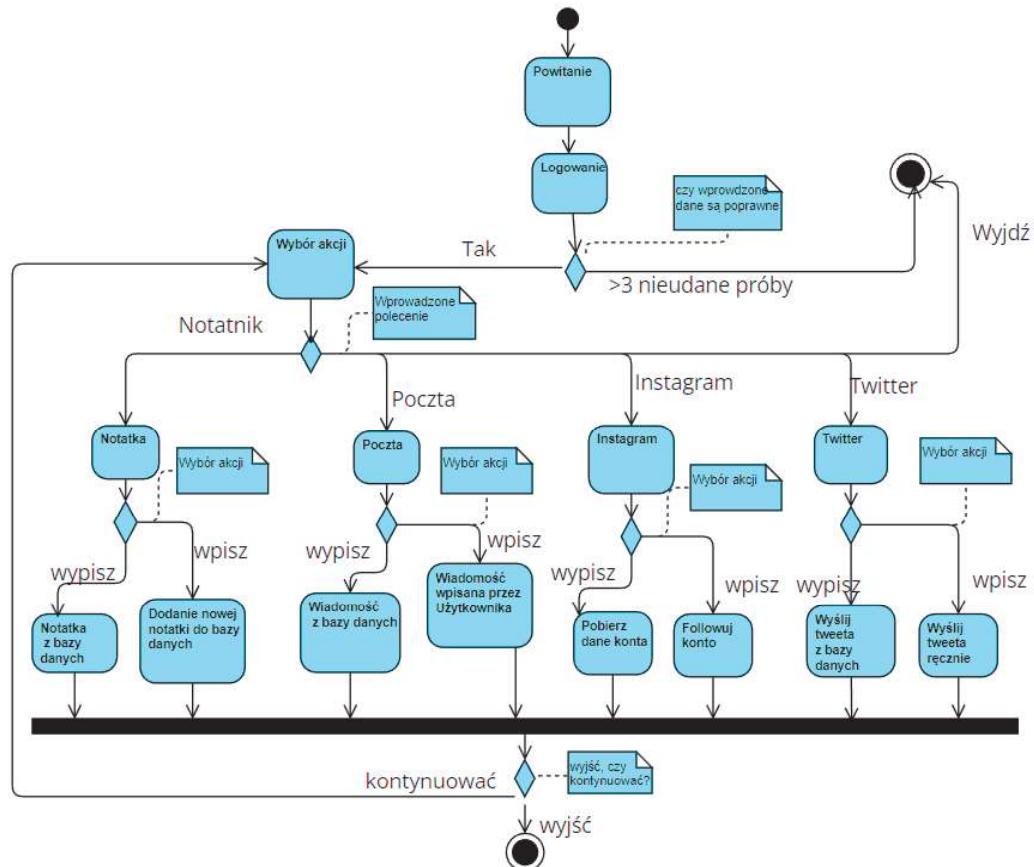
Rys. 10.1. Diagram ERD

11. Diagram klas



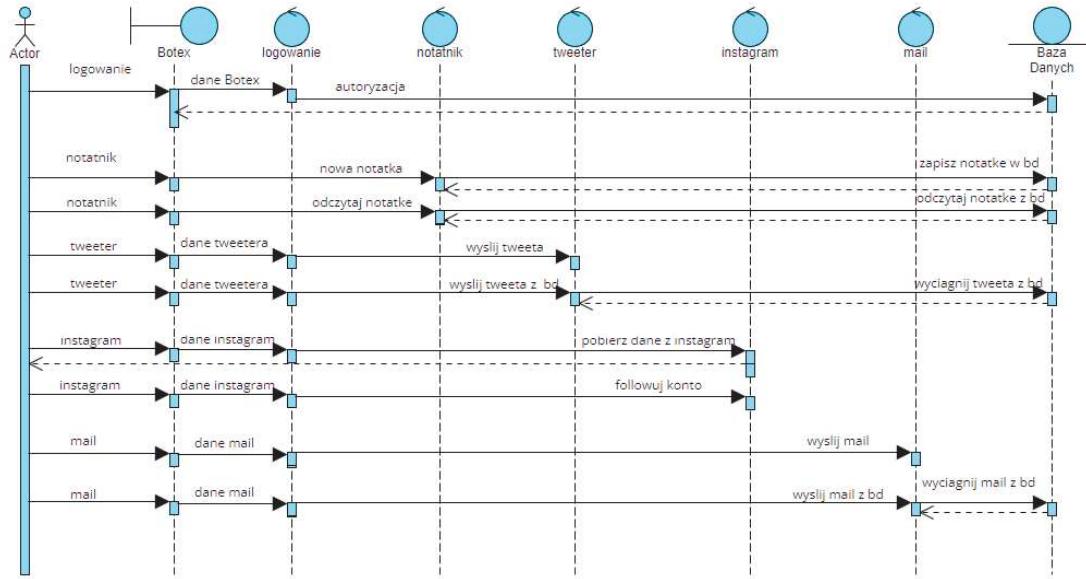
Rys. 11.1. Diagram klas

12. Diagram aktywności



Rys. 12.1. Diagram aktywności

13. Diagram sekwencji



Rys. 13.1. Diagram sekwencji

14. Rozpoznanie wzorców projektowych

- Wzorzec MVVM

Jego pełna nazwa brzmi Model – View – ViewModel. Całość opiera się na wydzieleniu trzech logicznych warstw w systemie. Celem MVVM jest wydzielenie odpowiednich warstw, aby zmniejszyć zależności. Każda z nich ma narzucone zasady, które powinna implementować, aby te zależności były jak najsłabsze.

Warstwa modelu jest to najniższa warstwa, posiada ona w sobie klasy, które odzwierciedlają domeny (czyli schemat danych naszej aplikacji). Powinny one jedynie odzwierciedlać dane, nie powinny posiadać w sobie żadnej logiki.

Kolejna warstwa to Widok, czyli odpowiada wizualnej stronie naszej aplikacji oraz prezentacji danych. Widok ma referencję do modelu widoku.

Ostatnia warstwa jest warstwa modelu widoku. Jej zadaniem jej udostępnienie danych dla widoku oraz wymianę informacji z modelem. To oznacza pobieranie danych oraz jego aktualizację. Sam model widoku udostępnia i zawiera tylko to, czego potrzebuje widok. Jest to skrojony pod niego dostawca danych.

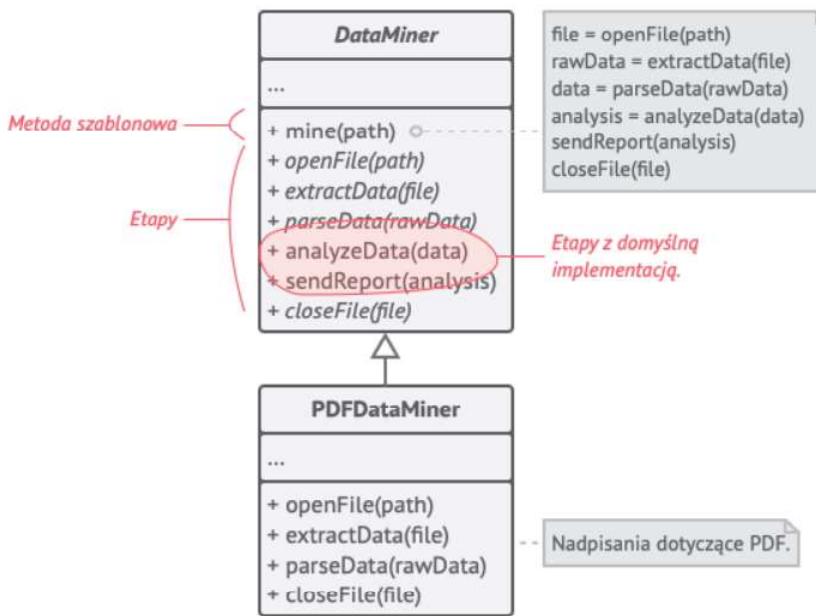
- Metoda Szablonowa

Metoda szablonowa to behawioralny wzorzec projektowy definiujący szkielet algorytmu w klasie bazowej, ale pozwalający podklasom nadpisać pewne etapy tego algorytmu bez konieczności zmiany jego struktury.

Metoda Szablonowa jest metodą, która zakłada rodzielenie algorytmu na kolejne etapy, utworzenie z nich metod, a także umieszczenie ciągu wywołań poszczególnych metod w jednej metodzie szablonowej. Etapy te dzielimy na abstrakcyjne, a takie, które posiadają domyślną implementację. Źeby móc skorzystać z algorytmu, klient powinien dostarczyć swoją podklasę - implementującą wszystkie etapy abstrakcyjne i nadpisać opcjonalne etapy jeśli jest taka potrzeba.

Na początek możemy zadeklarować wszystkie etapy jako abstrakcyjne, zmuszając podklasy do ich zaimplementowania. Jeżeli podklasy posiadają już konieczne implementacje, jedną rzeczą jaka jest konieczna do zrobienia jest dostosowanie sygnatur metod tak, aby zgadzały się z metodami klasy bazowej.

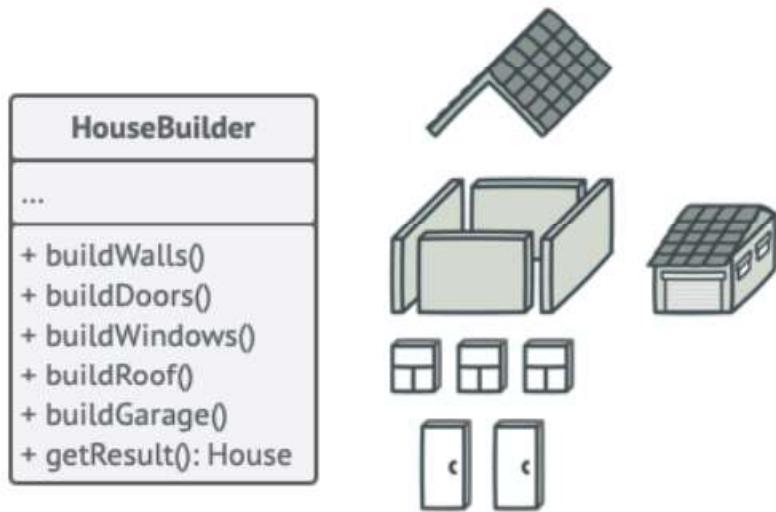
Istnieje jednak jeszcze jeden rodzaj etapów, zwane hookami. Hook to opcjonalny, pusty etap. Metoda szablonowa będzie działać nawet jeśli nie nadpisze się hooków. Zazwyczaj umieszcza się je przed i po istotnych etapach algorytmu, dając tym samym podklasom dogodne punkty zaczepienia, mogące służyć rozbudowie algorytmu.



Rys. 14.1. Metoda szablonowa dzieli algorytm na etapy, umożliwiając podklasom ich nadpisywanie, ale nie samą metodę szablonową.

- Wzorzec Budowniczy

Zamiast tworzyć podklasy, lub tworzyć konstruktory osobno dla każdego elementu (np dom z garazem, dom z ogrodkiem), tworzymy klasę który w swoich metodach posiada tworzenie każdego elementu krok po kroku (w przypadku domu, osobno utworzenie scian, sufitu itd). Dzieli konstrukcję obiektu na etapy, przy czym nie musimy korzystać z każdego etapu tworzenia, tylko z niezbędnych potrzebnych do utworzenia obiektu. Jeżeli obiekty różnią się u podstaw, ale etap ich tworzenia dalej jest taki sam, można utworzyć osobną klasę, nazwaną kierownikiem. Kierownik umożliwia ukrycie kodu konstrukcyjnego (budowniczego) przed klientem, wymianę budowniczego w trakcie tworzenia.



Rys. 14.2. Wzorzec Budowniczy pozwala konstruować złożone obiekty krok po kroku. Budowniczy ponadto nie pozwala na dostęp do nich innym obiektom, dopóki nie zostaną ukończone.

- Adapter

Adapter jest strukturalnym wzorcem projektowym pozwalającym na współdziałanie ze sobą obiektów o niekompatybilnych interfejsach.

Adapter jest specjalnym obiektem, który konwertuje interfejs jednego z obiektów w taki sposób, że drugi obiekt "go rozumie".

Możemy powiedzieć, że adapter jest swego rodzaju opakowaniem dla obiektu,

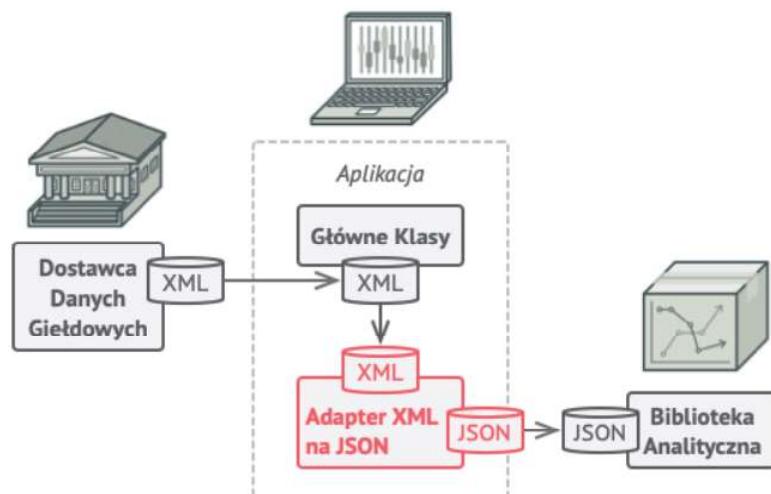
które ukrywa szczegóły konwersji. Obiekt, który jest opakowywany, może nawet nie wiedzieć o istnieniu adaptera. Dla przykładu: możemy np. opakować obiekt, który korzysta z jednostek kilometr i metr - w adapterze - konwertujący te dane na jednostki takie jak stopy i mile.

AdAPTERY są w stanie nie tylko konwertować dane pomiędzy formatami, ale także pozwolić na współpracę obiektów o różnych interfejsach.

Po krótkie możemy to opisać w następujący sposób:

1. Adapter uzyskuje interfejs kompatybilny z interfejsem jednego z obiektów.
2. Za pomocą tego interfejsu, istniejący obiekt może śmiało wywoływać metody adaptera.
3. Otrzymawszy wywołanie, adapter przekazuje je dalej, ale już w formacie obsługiwany przez opakowany obiekt.

Czasami jest nawet możliwe stworzenie adaptera dwukierunkowego, potrafiącego konwertować wywołania w obu kierunkach.



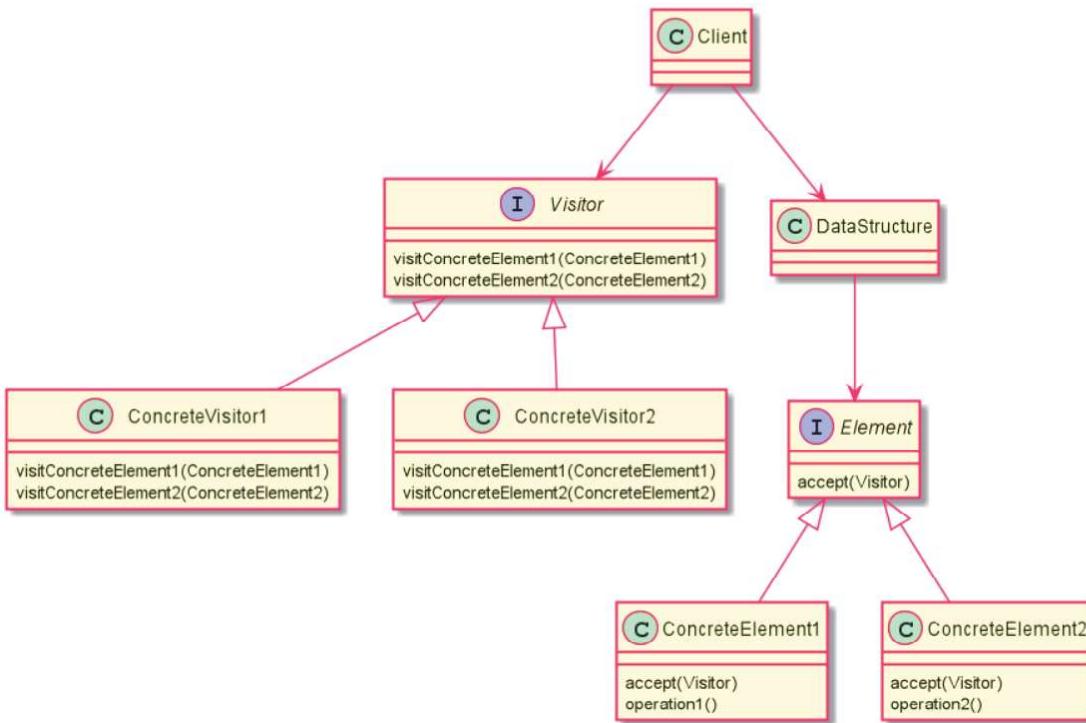
Rys. 14.3. Adapter jako wzorzec projektowy

- Wzorzec Singleton

Ogranicza tworzenie obiektów klasy do jednej instancji oraz zapewnienie globalnego dostępu do tej instancji. Umożliwia kontrolę nad współdzielonym zasobem. Tworzona instancja obiektu przechowywana jest w prywatnym lub chronionym statycznym polu, do którego dostęp poprzez referencje udziela metoda (konstruktor) który jako jedyny ma dostęp do instancji.

- Wzorzec Odwiedzający

1. Cel: Wzorzec Odwiedzający (visitor) to świetny sposób na oddzielenie algorytmu od struktury obiektu. Klasa Visitor zawiera zmiany i specjalizacje zamiast zmiany oryginalnego obiektu.
2. Problem: Chcemy mieć klasę ale osiągnąć nowych funkcji chcemy mieć w jednym miejscu. Mamy grupę obiektów i operację którą chcemy na nich wykonać. Logikę obiektów chcemy mieć w osobnej klasie.
3. Rozwiązanie: Wzorzec Odwiedzający (visitor) implementuje interfejs, który obsługuje każdy element i klasę wizytator dla niego. Wizytator wie o każdym elemencie, dla którego jest zaimplementowany. Zamiast zmieniać każdy element mamy klasę wizytatora którą zmieniamy.
4. Diagram klas wzorca Odwiedzający (visitor):



Rys. 14.4. Wzorzec Odwiedzający

- Wzorzec Dekorator

Umożliwia utworzenie specyficznego elementu wykorzystując dziedziczenie klasy głównej i dodaniu wymaganych zmian. Tworząc obiekty, przekazujemy je do konstruktorów w klasach dziedziczących tworząc wymaganą przez nas strukturę, łącząc istniejące dekoratory rozszerzając klasy o dodatkowe funkcjonalności.

Dekorator

implementuje ten sam interfejs co klasa bazowa, z punktu widzenia klienta są identyczne.

- Fasada

Fasada jest strukturalnym wzorcem projektowym, który wyposaża bibliotekę, framework lub inny złożony zestaw klas w uproszczony interfejs.

Fasada to klasa stanowiąca prosty interfejs dla złożonego podsystemu, zawierającego mnóstwo ruchomych części. Fasada może dawać ograniczoną funkcjonalność, w porównaniu z korzystaniem z elementów podsystemu bezpośrednio, ale za to eksponuje tylko te możliwości, których klient naprawdę potrzebuje.

Stworzenie fasady jest wygodnym sposobem integracji twej aplikacji ze skomplikowaną biblioteką posiadającą wiele funkcji, gdy potrzebujesz tylko wąskiego zakresu jej funkcji.

- Wzorzec Kompozyt

Przedstawienie klas w postaci drzewa, z jedną klasą nadrzędną która w swoim interfejsie umożliwia kontrolowanie obiektów-gałęzi. Wzorzec wyróżnia elementy: Component - klasa abstrakcyjna reprezentująca pojedyńcze obiekty leaf, jak i kontenery tych obiektów Leaf - typ prosty, bez potomków Composite - przechowuje obiekty proste (leaf), implementuje zachowanie elementów które zawiera

Composite i leaf dziedziczą ten sam interfejs, co pozwala na dostęp do obiektów prostych w ten sam sposób jak do grupy tych obiektów. Operacje klienta mogą zostać wykonane na pojedyńczym obiekcie jak i na grupie obiektów.

- Wzorzec obserwator

Klasa publikująca zbiera w sobie liste osób którzy chcą zostać powiadomiani w przypadku wystąpienia jakiegoś wydarzenia. Za każdym razem klasa publikująca, gdy wydarzy się coś ważnego, przejrzy listę swoich subskrybentów a następnie ich powiadomi. Aplikacje mogą mieć duzo różnych klas służących do powiadomiania dla jednego konkretnego wydarzenia. Wszyscy subskrybenci powinni implementować ten sam interfejs i publikujący powinien komunikować się wyłącznie przez ten interfejs. W interfejsie powinien być zadeklarowana metoda powiadamiania, z zestawem parametrów za pomocą których publikujący może przekazać dodatkowe dane kontekstowe wraz z powiadomieniem.