**Uniwersytet Warmińsko-Mazurski w Olsztynie**

**Wydział Matematyki i Informatyki**

**Zespół TSR:**

**Damian Tomczak**

**Dominik Słodkowski**

**Bartosz Rybiński**

**Block-based programming language**

Praca zaliczeniowa napisana

napisana pod kierunkiem

mgr inż. Tomasza Żmijewskiego

Olsztyn 2022

Spis treści

[1 Opis firmy 4](#_Toc121318189)

[2 Słownik Pojęć 6](#_Toc121318190)

[3 Zakres organizacji, dziedziny problemowej, aktorzy 7](#_Toc121318191)

[4 Model Biznesowy 8](#_Toc121318192)

[4.1 Biznesowy Diagram Przypadków Użycia 8](#_Toc121318193)

[4.2 Aktorzy 8](#_Toc121318194)

[4.3 Przypadki użycia 8](#_Toc121318195)

[4.3.1 Ucz się w programie 8](#_Toc121318196)

[4.3.2 Ranking 10](#_Toc121318197)

[4.3.3 Administrowanie systemem 10](#_Toc121318198)

[4.3.4 Zarządzaj kadrą 10](#_Toc121318199)

[4.3.5 Zarządzaj materiałami edukacyjnymi 11](#_Toc121318200)

[5 Model systemowy 14](#_Toc121318201)

[5.1 Systemowy diagram przypadków użycia 14](#_Toc121318202)

[5.2 Aktorzy 14](#_Toc121318203)

[5.3 Przypadki Użycia 14](#_Toc121318204)

[5.3.1 Wybierz Zadanie 14](#_Toc121318205)

[5.3.2 Buduj Program 15](#_Toc121318206)

[5.3.3 Kompiluj program 15](#_Toc121318207)

[5.3.4 Oceń zadanie 16](#_Toc121318208)

[5.3.5 Dostosuj program 16](#_Toc121318209)

[5.3.6 Autoryzuj użytkownika 17](#_Toc121318210)

[6 Dokumenty występujące w ramach dziedziny problemowej 18](#_Toc121318211)

[7 Lista klas, atrybutów, związków 22](#_Toc121318212)

[8 Diagram klas – konceptualny 23](#_Toc121318213)

[9 Diagram Klas - implementacyjny 24](#_Toc121318214)

[10 Diagram obiektów 25](#_Toc121318215)

[11 Lista funkcji 26](#_Toc121318216)

[12 Aspekt UX 28](#_Toc121318217)

[12.1 Typ interfejsu 28](#_Toc121318218)

[12.2 Wymagane urządzenia 28](#_Toc121318219)

[12.3 Grupy użytkowników 28](#_Toc121318220)

[12.4 Typowe zadania 28](#_Toc121318221)

[12.5 Scenariusze do zadania 28](#_Toc121318222)

[12.6 Projekty Ekranów 29](#_Toc121318223)

[12.6.1 Szablon 29](#_Toc121318224)

[12.6.2 Ekran wyboru zadań 30](#_Toc121318225)

[12.6.3 Ekran schematu z konsolą 31](#_Toc121318226)

[12.6.4 Ekran schematu z rozwijaną listą 31](#_Toc121318227)

[12.6.5 Ekran schematu łączenia bloków 32](#_Toc121318228)

[12.6.6 Ekran schematu z komunikatem o wykonaniu zadania 33](#_Toc121318229)

[13 Spis Diagramów 34](#_Toc121318230)

[14 Spis Ilustracji 34](#_Toc121318231)

# Opis firmy

TSR School jest franczyzodawcą o zasięgu ogólnokrajowym, który swoje placówki ma zlokalizowane w średnio zaludnionych polskich aglomeracjach powyżej 50 tyś mieszkańców, oraz większych miastach. Poza tym firma dostarcza narzędzia do nauczania zdalnego w języku angielskim oraz kursy online, jak również ‘bootcampy’ programistyczne i szkolenia z zakresu kompetencji cyfrowych.

Przedsiębiorstwo jest akademią IT z prawie 10-letnim doświadczeniem proponującym nieszablonowane rozwiązania w zakładaniu innowacyjnego biznesu w sektorze edukacyjnym. Korzystając z doświadczenia, „know-how”, materiałów edukacyjnych i platform do zarządzania.

Główna siedziba firmy zlokalizowana jest w Olsztynie na ulicy Słonecznej 54, 10-561 i otwarta jest w godzinach 8-16 (pomoc telefoniczna dla franczyzobiorców oraz student jest otwarta w godzinach pracy placówki). Preferowana metoda komunikacji ze szkołą odbywa się za pośrednictwem poczty elektronicznej pod adresem [contact@tsr.edu.pl](mailto:contact@tsr.edu.pl).

Firma zajmuje się nauką programowania i posiada system zarządzający przydzielaniem i oceną zadań. Umożliwia on autoryzację użytkownika oraz komunikację ze studentami i nauczycielami. Zadania są realizowane przez uczniów w postaci napisanego kodu, który jest wykonywany na urządzeniu użytkownika a informacja o wykonaniu zadania zostaje przekazana do systemu zarządzającego gdzie jest to zapisywane po stronie serwera firmy. Efekty działania oceniane są poprzez system rankingowy, posiadający publicznie dostępną tablicę wyników z najlepszymi studentami w danej grupie.

System to cała platforma nauczania, która oprócz przydzielania i oceny zadań dostarcza narzędzia do nauczania dla poszczególnych grup w zależności od zaznajomienia się z informatyką.

Istnieje luka w systemie nauczania którą TSR School chce załatać. Nauka programowania za pomocą kodu jest przystępna dla osób, które miały już styczność z komputerami oraz znają język angielski. Problemem są dzieci i osoby o niskich zdolnościach cyfrowych, w którym wypadku nauczanie przebiegało za pomocą pisania i analizowania skryptów w pythonie. Osoby te, które dopiero uczą się pisać na klawiaturze, nie są w stanie efektywnie pisać kodu.

Z komentarzy osób prowadzących oraz uczniów pilotażowej grupy uczącej się poprzez pisanie skryptów pythona wynikało, że materiał jest zbyt skomplikowany. Pierwsze 3 zajęcia poświęcali na naukę wprowadzania danych do systemu, wizualizacja za pomocą rysunków nie była wystarczająca dla uczniów, a język angielski wprowadzał dodatkowe obciążenie dla studentów.

Firma podejmowała się, również prób nauki poprzez fizyczne obiekty służące za analogi elementów wykorzystywanych do stworzenia programu:

* drewniane bloczki o różnych kształtach – podstawowe operacje warunkowe oraz arytmetyczne, pętle.
* karteczki samoprzylepne – zmienne
* deska do krojenia – podstawa programu, funkcji.

Ta metoda nauki, wymagała dużego zaangażowania ze strony wysoko płatnych prowadzących. Wykonywali oni kompilacje w głowie żmudnie analizując krok po kroku program napisany przez ucznia w trakcie zajęć. Brakowało również ujednoliconej składni, oraz informacji o błędach które normalnie świadczą o złym działaniu programu. To wszystko przy jednoczesnym pilnowaniu uczniów którzy ze względu na młody wiek, zachowywali się często niestosownie.

Forma nauki poprzez drewniane klocki analogiczne do elementów programu była atrakcyjna dla uczniów, rozwiązanie to jednak było nieefektywne. Wymagania wobec prowadzących początkowo skutkowały wymogiem posiadania więcej niż jednego nauczyciela przy realizacji zajęć grup najsłabszych pod względem umiejętności cyfrowych i programistycznych.

Spowodowało to znaczący wzrost kosztów operacyjnych firmy, która w ostatnim kwartale zanotowała niższy niż przewidywany wzrost zysków. Zmusiło to do wycofania się z przystępnej formy nauki na rzecz mniej intuicyjnych klasycznych rozwiązań programistycznych. Firma jest jednak świadoma luki w obszarze nauczania programowania i przewiduje wzrost konkurencji w następujących latach. Sprawia to, że jest zainteresowana rozwojem nowatorskich narzędzi nauczania.

Po konsultacjach z firmą okazało się, że bardzo dobrym rozwiązaniem, który pokryłby lukę w nauczaniu w prosperującej gałęzi nauczania niezaznajomionych z technologią, jest narzędzie, które przeniosłyby poprzednie doświadczenie z grupą testową do systemu informatycznego oraz bazując na badaniach uniknęłoby wcześniej wymienionych wzrostów kosztów operacyjnych lecz nadal przybliżałoby te osoby do obsługi komputera. Okazało się również, że pomimo początkowej łatwości w poruszaniu się podczas zabawy w języku ojczystym dla studentów, podczas dalszych etapów nauki stanowiło to problem, dlatego interfejs użytkownika powinien być w języku angielskim.

# Słownik Pojęć

Użytkownik = Osoba korzystająca z systemu zapisana w bazie danych, posiada login i hasło.

Student = Użytkownik zapisany w rankingu posiadający możliwość budowania schematów oraz wybierania i rozwiązywania przydzielonych zadań.

Nauczyciel = Użytkownik posiadający możliwość przydzielania zadań dla studentów, mający dostęp do schematów studentów oraz możliwość monitorowania postępu ich nauki.

Blok = Podstawowa część schematu, mogąca realizować tworzenie i zmianę zmiennych, operacje warunkowe i arytmetyczne oraz pętle warunkowe, mogący wchodzić z interakcję z innymi blokami.

Schemat = Algorytm stworzony za pomocą bloków i relacji między nimi, którego wynik wypisywany jest na konsole.

Ranking = Lista prezentująca postęp nauki studenta z porównaniem do innych studentów, na podstawie ukończenia zadań dla nich przydzielonych.

Zadanie = Przydzielony przez prowadzącego zajęcia problem algorytmiczny, który jest rozwiązywany budując schemat.

Konsola = Standardowe wyjście tekstowe w dostępnym systemie, na którym wyświetlane są wyniki działania schematu.

Platforma nauczania = Należące do firmy multifunkcjonalne oprogramowanie dostarczające systemowi możliwość autoryzacji, informacji o studentach jak przydzielonych do nich zadaniach oraz zestaw narzędzi do nauki programowania. Przechowuje też informacje o postępie nauki studentów.

System = Jedno z narzędzi do nauki programowania z którego korzysta student. Do jego działania niezbędne są informacje dostarczane przez system.

TODO: sprawdź słownictwo

* Organizacja – TSR/TSR School zwany również franczyzodawca zajmujący się organizacją usług w dziedzinie edukacji w branży IT
* Otoczenie organizacji – firmy rekrutujące specjalistów IT, inne szkoły programowania
* Intersariusze organizacji - dzieci i nastolatkowie, human reasources teams
* Dziedzina problemowa – nowoczesne technologie
* Intersariusze dziedziny problemowej - firmy technologiczne
* Pracownicy biznesowi – główna siedziba firmy, nauczyciele

# Zakres organizacji, dziedziny problemowej, aktorzy

Firma TSR School

Organizacja = cała firma

Struktura firmy = nauczyciel, placówka dydaktyczna, główna siedziba firmy

Dziedzina problemowa = nauka programowania i problem solving

Aktorzy biznesowi: student, nowy student

Pracownicy biznesowi: nauczyciele

PU:

- ~~Utwórz studenta~~

- ~~Przydziel zadanie~~ -> Udostępnianie danych

- ~~Autoryzuj użytkownika~~ -> Autoryzowanie

- ~~Wybierz zadanie~~ -> Nauka w programie

- ~~Buduj program~~ -> Nauka w programie

- ~~Kompiluj program~~ -> Nauka w programie

- ~~Oceń zadanie~~ -> Ranking

- ~~Dostosuj program~~ -> Nauka w programie

Aktorzy biznesowi: student, ~~nowy student~~

Pracownicy biznesowi: nauczyciel

PU:

- Administrowanie systemem

- Ranking

- Nauka w programie

Aktorzy biznesowi: student

Pracownicy biznesowi: nauczyciele

# Model Biznesowy

## Biznesowy Diagram Przypadków Użycia

Diagram

Description automatically generated

Diagram 1 Biznesowy Diagram Przypadków Użycia

## Aktorzy

* Student
* Główna siedziba firmy
* Nauczyciel
* Platforma nauczania

## Przypadki użycia

### Ucz się w programie

Student nabywa nowych umiejętności w programie, a jego obsługa sprowadza się do: dostosowaniu preferencji obsługi, wybieraniu przydzielanych zadań oraz tworzeniu skryptu za pośrednictwem bloków oraz sprawdzaniu i analizy wyników.

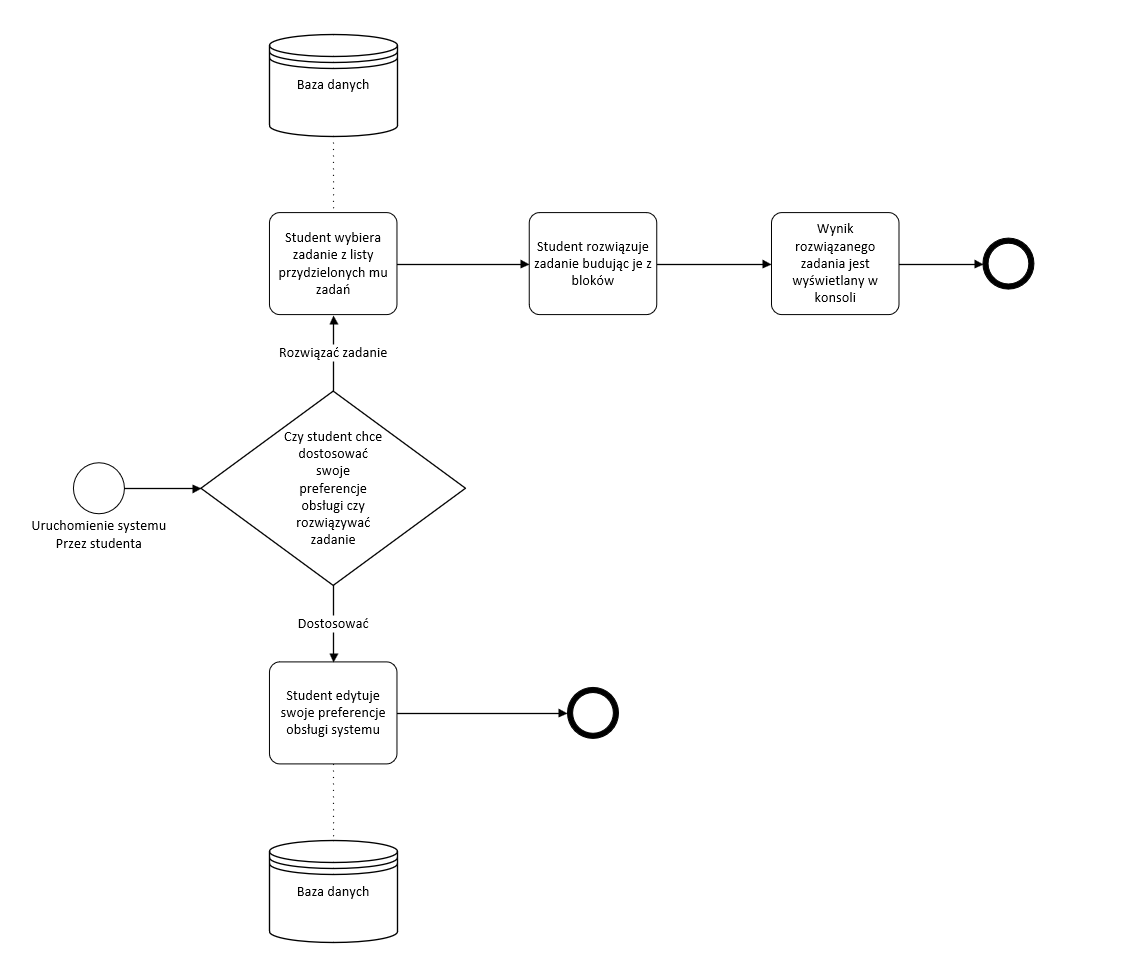


Diagram 2 BPMN ucz się w programie

TODO: grafika

1. Udostępnianie danych

Program do swojego działania potrzebuje możliwości autoryzacji użytkownika, pobierania zadań dla zalogowanego użytkownika.

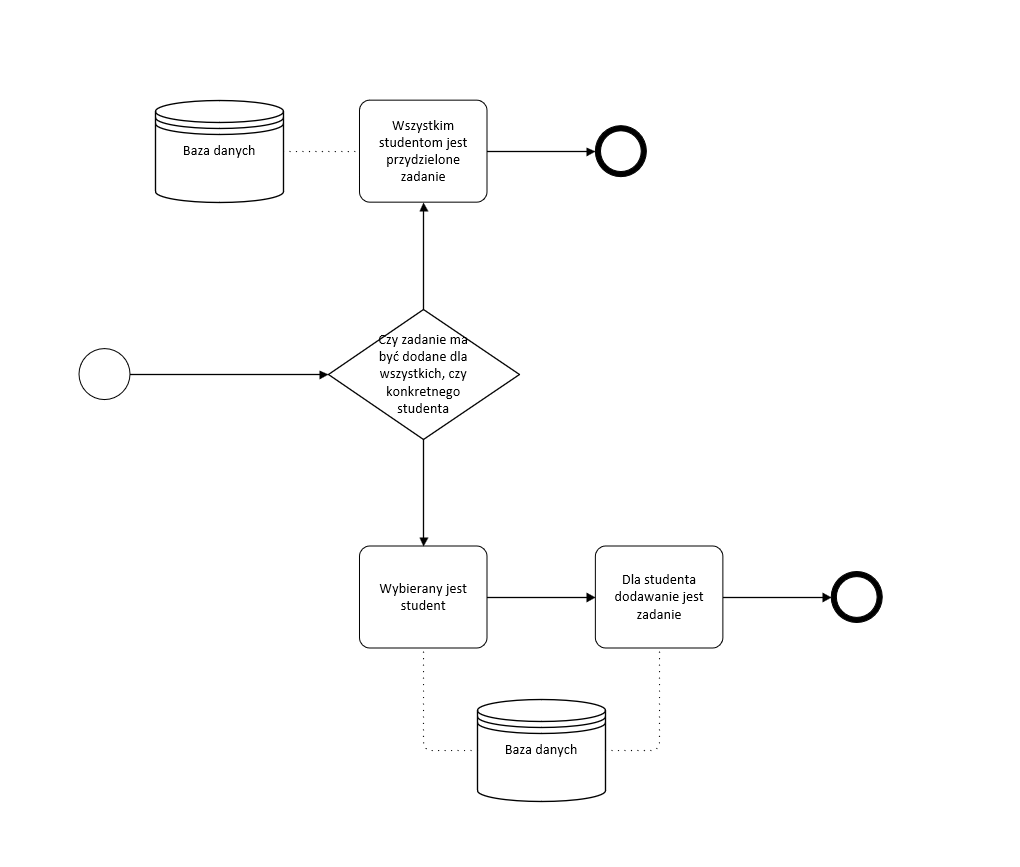


Diagram 3 BPMN opis

TODO: grafika

### Ranking

Student w systemie ma możliwość monitorowania swoich osiągów oraz motywowania się dysponując możliwością porównaniu swoich wyników z rówieśnikami z grupy. System dostarcza informacji o wyniku zadania, aby platforma nauczania mogła sprawdzić te dane i odnotować to studentowi i rankingowi.

TODO: grafika

### Administrowanie systemem

Nauczyciel w platformie nauczania monitoruje swoich studentów oraz grupę, zarządza dostępnymi zadaniami oraz ich przydziałem, ma możliwość tworzenia również zadań.

TODO: grafika

### Zarządzaj kadrą

Główna siedziby firmy umożliwia zakładanie placówek korzystających z ich brandingu, materiałów szkoleniowych. Monitoruje zadowolenie studentów oraz postępy placówek jak i pracowników.

TODO: grafika

### Zarządzaj materiałami edukacyjnymi

Główna siedziba firmy udostępnia materiały edukacyjne wchodzące w skład platformy nauczania.

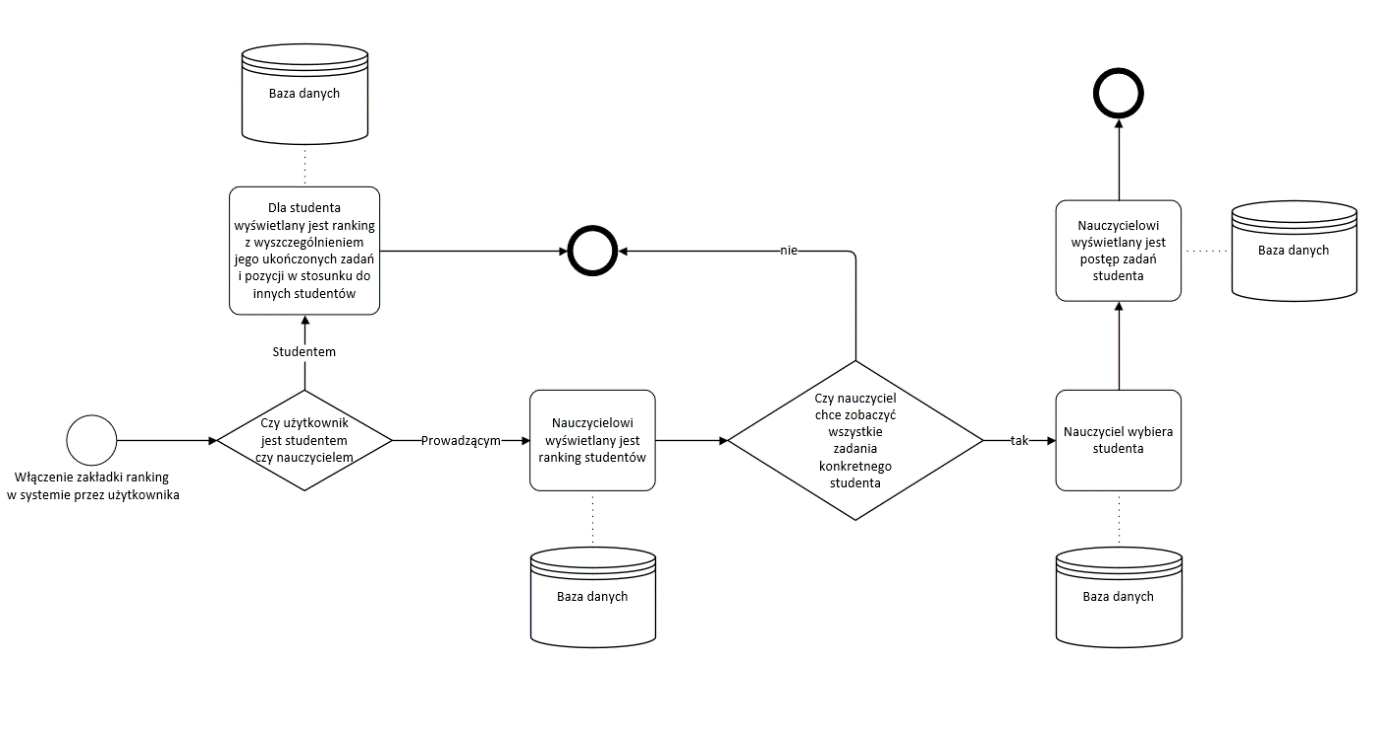


Diagram 4 BPMN Zarządzaj materiałami edukacyjnymi

TODO: przerób grafikę

1)

Przykład:

Student Milena Samsel wybiera dostępne dla niej zadanie o nazwie „dzielenie z resztą”a następnie rozwija bazowy schemat składających się z dwóch bloków typu zmienna o wartościach 2 i 3 o odpowiednie bloki, aby osiągnąć wymieniony cel – „osiągnij resztę z dzielenia 5”. Przyciskiem Start student prosi system o wykonanie schematu, następnie program prezentuje na konsoli wynik działania schematu.

Specyfikacja:

- student wybiera zadanie

- student rozwija schemat

- program wykonuje schemat

- program prezentuje wynik działania schematu

2)

Przykład:

Student Milena Samsel próbuje się zalogować w programie. Dane uwierzytelniania (milena.samsel1 password123) są przetwarzane przez system. Zalogowanie kończy się prawidłowo, program następnie prosi system o dostarczenie zadań, które zostały przypisanego do wykonania dla studenta: *https://www.gov.pl/tasks/task1.*json (nazwa zadania to *dzielnie i mnożenie*) #id0 *https://www.gov.pl/tasks/task2.json* (nazwa zadania to *dodawanie i odejmowanie*) #id1.

Specyfikacja:

- autoryzacja użytkownika

- przypisanie zadań

3)

Przykład:

Student Milena Samsel uruchamia wykonanie schematu dla zadania *dzielenie i mnożenie.* Program przekazuje wynik działania schematu (5) do Systemu, gdzie jest on porównywany z pożądanym wynikiem 5. Wynik jest prawidłowy, dlatego informacja o prawidłowym wykonaniu zadania wraca do programu. A prawidłowe wykonywanie zadania dla studenta jest zapisywane w rankingu.

Specyfikacja:

- sprawdzenie wyniku

- aktualizacja rankingu

4)

Przykład:

Tydzień przed rozpoczęciem semestru zimowego 2022-2023 nauczyciel Zbigniew Zengota otrzymuje listę studentów przypisanych do jego grupy (milena.samsel1, jarek.warchulski1, weronika.traba1, dominik.solniski1). Tworzy im konta po czym przypisuje po trzy przykładowe zadania (#1 dodawanie, #2 odejmowanie, #3 reszta z dzielenia); Po miesiącu sprawdza postęp wykonania zadań przez studentów.

Specyfikacja:

- tworzenie zadań

- przydzielanie zadań

- monitorowanie postępów

- tworzenie kont studentów

5)

Przykład:

Nowa placówka w Ostrołęce jest zainteresowana podjęciem współpracy z firmą. Po dokonaniu niezbędnych formalności(umowa franczyzowa). Placówce udostępniany jest dostęp do platformy nauczania oraz materiałów szkoleniowych(„jak dbać o studenta”). Pracownicy Zbigniew Zengota oraz Jarosław Chrostek są szkoleni z korzystania z platformy nauczania. Pod koniec semestru główna siedziba firmy prowadzi test zadowolenia ze szkoły (wynik to 95%).

Specyfikacja:

- tworzenie nowej placówki

- udostępnianie materiałów szkoleniowych

- szkolenie kadry pracowniczej

- monitorowanie placówek

6)

Przykład:

W wyniku zapotrzebowania rynku, bazując na przeprowadzonych badaniach i poprzednim doświadczeniu, główna siedziby firmy wprowadza nowy system przystosowany dzieciom i młodzieży.  
Po miesiącu od wprowadzenia nowego systemu wykryto w nim błąd (komunikat błędu 666), który uniemożliwiał dalszą pracę, potrzebny fix (potrzebna była zmiana systemu odbioru danych o zadaniach, po zmianie w bazie danych).

Specyfikacja:

- dodanie nowego systemu nauczania

- utrzymanie platformy nauczania

# Model systemowy

## Systemowy diagram przypadków użycia

Diagram

Description automatically generated

Diagram 5 Systemowy Diagram Przypadków Użycia

## Aktorzy

* Student
* Platforma nauczania

## Przypadki Użycia

### Wybierz Zadanie

Student po zalogowaniu i pobraniu przez program informacji o podzielonych do niego zadań widzi ekran z możliwymi do wyboru zadaniami przypisanymi mu przez nauczyciela. Wybiera konkretne zadanie i przechodzi do budowania programu.

* + - 1. Wykonaj 6. Autoryzuj użytkownika
      2. Wykonaj 7. Przydziel zadania
      3. System wyświetla listę zadań dostępnych dla studenta

2a. System wyświetla komunikat o braku dostępnych zadań

Zakończ

* + - 1. Student wciska przycisk „wybierz”
      2. Wykonaj 2. Buduj Program

TODO: grafika

Studentowi Karolowi Tuskowi po zalogowaniu system prezentuje wertykalną listę przypisanych do jego zadań w postaci identyfikator oraz nazwa (#1 Dzielenie z resztą, #2 Dodawanie, #3 Równanie kwadratowe). Karol zaznacza zadanie o identyfikatorze #2 i wybiera wciska przycisk „wybierz”.

### Buduj Program

Student za pomocą bloków(zestaw instrukcji bądź instrukcja do wykonania przez procesor) które są układane przez niego w wybranej kolejności tworzy algorytm po czym może wcisnąć przycisk Start.

* + - 1. System wyświetla planszę budowania
      2. Student wyświetla listę bloków

2a Student postanawia opuścić zadanie

Wykonaj 1. Wybierz zadanie

* + - 1. Student wybiera blok z listy
      2. System umieszcza blok na planszy
      3. Student modyfikuje blok

4a. Student modyfikuje wartość bloku

4b. Student modyfikuje wyjście/a bloku

4c. Student modyfikuje wejście/a bloku

* + - 1. Student podejmuje decyzję

5a. Student dalej buduje program

Wróć do 5.1. System wyświetla planszę budowania

5b. Student wciska przycisk „start”

Wykonaj 3. Kompiluj program

TODO: grafika

Student Karol Tusk po zalogowaniu i wybraniu zadania „#2 Dodawanie” wyświetlana jest przez system plansza do budowania. Użytkownik zapoznaje się z istniejącymi blokami (blok typu zmienna int z wartością 1, blok typu zmienna int z wartością 2 oraz blok output) oraz celem działania programu „uzyskaj wynik na wyjściu 3”. karol.tusk1 następnie otwiera menu wyboru bloków i z kategorii działania arytmetyczne wybiera addition. Kolejną czynnością studenta jest połączenie wejść bloku addition do zmiennych typu int wartością 2 oraz 1, wyjście bloku addition łączy z blokiem output. Aby sprawdzić rezultat student wciska przycisk „start”.

### Kompiluj program

Po wciśnięciu przycisku start program blokowy jest parse’owany do pythona i jest interpretowany przez interpreter. Jeżeli program jest napisany poprawnie jego wynik jest wyświetlany studentowi na konsoli, jeżeli nie zwracany jest błąd.

* + - 1. System parsuje schemat blokowy na kod pythona
      2. System wykonuje kod pythona

2a. System zwrócił błąd podczas interpretowania kodu

2a.1. System wyświetla błąd w konsoli

2a.2. Wykonaj 2. Buduj program

* + - 1. System wyświetla wynik działania kodu na konsoli
      2. Wykonaj 4. Oceń zadanie

TODO: grafika

System otrzymuje listę bloków w aplikacji wraz z ich stanem i relacjami miedzy nimi. Bloki to zmienna typu int z wartością 2, zmienna typu int z wartością 1, operacja arytmetyczna addition, blok output. Wejścia bloku addition zawierają relacje z blokami zmiennych z wartościami 1 oraz 2, natomiast wyjście zawiera relacje z outputem. System parsuje schemat na kod pythona. Wynik jego działania jest prawidłowy i wyświetla komunikat 3. Komunikat z wartością 3 jest prezentowany na konsoli studenta i przekazywany do oceny przez system.

### Oceń zadanie

System po wykonaniu schematu porównuje rezultat z oczekiwanym rezultatem potem wyświetla stosowny komunikat w zależności od wyniku porównania, a uzyskanie prawidłowego zapisuje do bazy.

* + - 1. System odbiera wynik
      2. System sprawdza wynik

2a. Jeśli wynik prawidłowy

2a.1. System aktualizuje wykonanie zadanie

2a.2 Program wyświetla komunikat o prawidłowym wykonaniu na konsoli

2a.3 Wykonaj 2. Buduj program

2b. Jeśli wynik nieprawidłowy

2b.1 Program wyświetla komunikat o nieprawidłowym wykonaniu na konsoli

2b.2 Wykonaj 2. Buduj program

System odbiera wynik działania schematu studenta Karola Tuska w postaci stringu „3”, jego wartość jest porównywana z oczekiwaną wartością przypisaną do zadania „3”. Wynik to prawda, dlatego na konsoli studenta pojawia się komunikat o prawidłowym wykonaniu zadania, a wykonanie zadania rejestrowane jest w bazie danych.

TODO: grafika

### Dostosuj program

Użytkownik i prowadzący ma możliwość podstawowej możliwości dostosowania programu do własnych preferencji za pośrednictwem paska zadań (m.in. tryb nocny).

* + - 1. System odbiera żądanie personalizacji
      2. System obsługuje żądanie personalizacji

Użytkownik Jarosław Kaczyński naciska przycisk „inne” na pasku zadań, system rozwija opcje ukryte spod tej opcji, które składają się z informacji o programie i pomocy. Następnie użytkownik naciska opcję „pomoc”. System prezentuje mu pomoc w obsłudze programu. Użytkownik zapoznaje się i zamyka okno.

Diagram

Description automatically generated

Diagram 6 TODO: OPIS

### Autoryzuj użytkownika

Student wpisuje login I hasło w aplikacji. Jeżeli wprowadził prawidłowe dane przechodzi do przeglądu zadań. Jeśli wpisał złe informacje proszony jest o ponowne wpisanie hasła i loginu. Wprowadzone dane są później wykorzystywane w celu weryfikacji dostępu do danych.

1. Przydziel Zadania

Nauczyciel tworzy nowe zadanie dla studentów w systemie z przykładowymi niekompletnymi rozwiązaniami, po czym umieszcza je w bazie danych i przypisuje wybrane z bazy danych zadania wybranym studentom. Następnie o te dane prosi program podczas pracy.

* + - 1. Program prosi o dane przypisane dla zalogowanego studenta
      2. System zwraca przypisane zadania dla zalogowanego studenta

TODO: grafika

Program z zalogowanym studentem Stanisławem Kowalskim o identyfikatorze #666 prosi o przypisane dla niego zadania. System zwraca, że przypisane dla niego zadania znajdują się pod adresami *https://www.gov.pl/tasks/task1.*json (nazwa zadania to *dzielnie i mnożenie*) *https://www.gov.pl/tasks/task2.json* (nazwa zadania to *dodawanie i odejmowanie*).

# Dokumenty występujące w ramach dziedziny problemowej

**Graphical user interface, text, application

Description automatically generated**

Rysunek 1 Formularz rejestracji

**formularz\_rejestracji** = rodzaj\_kursu + { dane\_uczestnika } + { dane\_opiekuna } + skad\_wiesz + uwagi

**rodzaj\_kursu** = [on-site | remote]

**dane\_uczestnika** = email + rok\_urodzenia + imie\_i\_nazwisko

**dane\_opiekuna** = email + nr\_telefonu + imie\_nazwisko + kod\_pocztowy

Graphical user interface, text, application

Description automatically generated

Rysunek 2 Dyplom

**dyplom** = imie\_nazwisko\_uczestnika + prowadzacy\_zajecia + data

Text

Description automatically generated

Rysunek 3 Oświadczenie RODO

**zgoda\_przetwarzania\_danych\_osobowych** = data + podpis



Rysunek 4 Faktura

**faktura\_za\_serwer** = data\_wystawienia + data\_wystawienia + nr\_faktury + nazwa\_sprzedawcy + adres\_sprzedawcy + nip\_sprzedawcy + vat\_ue + nabywca + adres\_nabywcy + płatnosc + zamowienie + pelna\_cena + uwagi

**platnosc** = [ kart--a | przelew | przelewy24 ]

**zamówienie** = { usluga }

**usluga** = nazwa\_serwera + ilosc + cena\_netto + vat + warto\_brutto

# Lista klas, atrybutów, związków

Klasy:

**Użytkownik**

* Metody komunikacji

**Student, dziedziczy po użytkowniku**

* Postęp nauki studenta

**Zadanie**

* Określony cel
* Zadane warunki początkowe
* Nazwa

**Blok**

* Rodzaj działania

**Połączenie**

* Wejścia
* Wyjścia

**Schemat**

* Bloki
* Warunki początkowe

# Diagram klas – konceptualny

**Diagram

Description automatically generated**

Diagram 7 Konceptualny Diagram Klas

# Diagram Klas - implementacyjny

-Diagram

Description automatically generated

Diagram 8 Implementacyjny Diagram Klas

# Diagram obiektów

Diagram

Description automatically generated

Diagram 9 Diagram Obiektów

[Uzasadnienie prawidłowości użycia link object](http://etutorials.org/Programming/Learning+uml/Part+II+Structural+Modeling/Chapter+3.+Class+and+Object+Diagrams/3.2+Associations+and+Links/)

# Lista funkcji

-wybór zadania

-przydziel zadania

-dodaj blok do schematu

-edytuj zawartość bloku

-edytuj połączenia

-sparsuj schemat na kod

-interpretuj kod

-wyświetl wynik na konsoli

-zweryfikuj wynik

-dostosuj program

-autoryzuj użytkownika

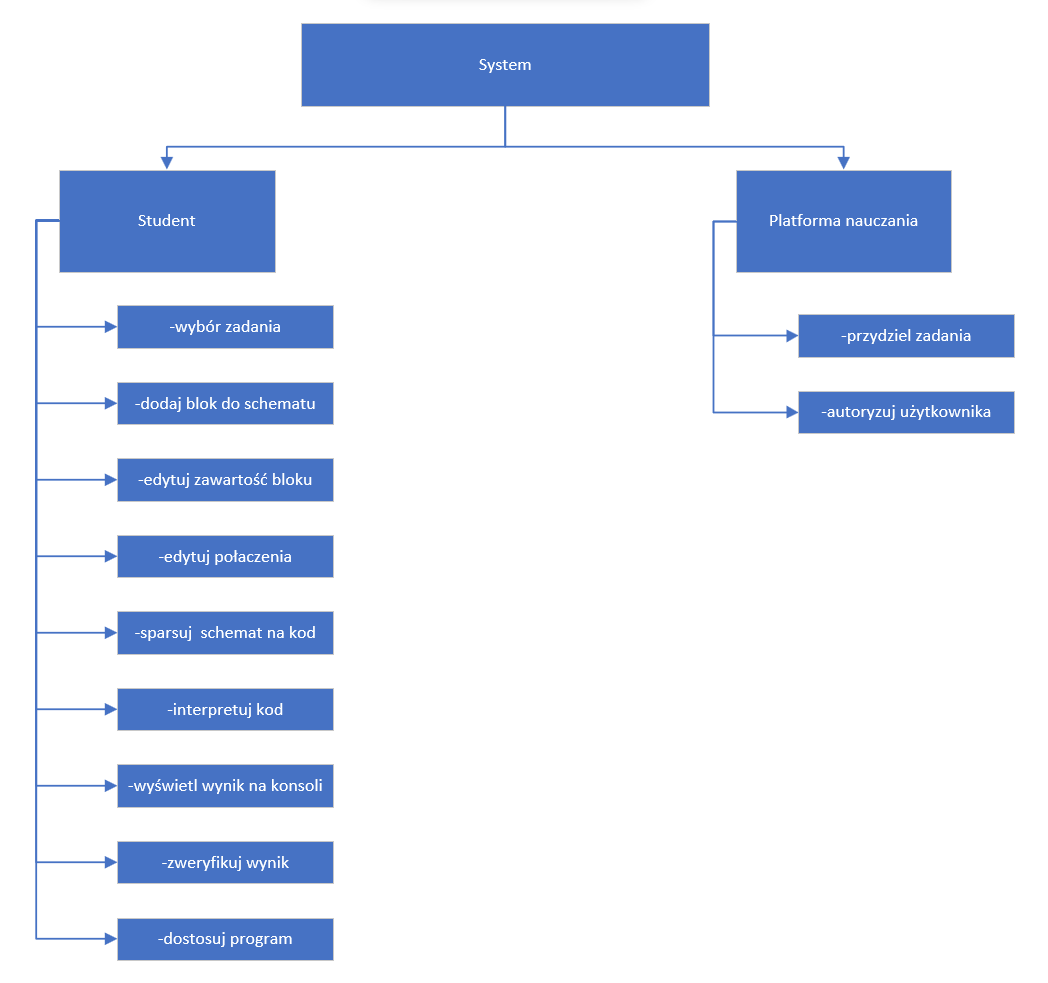


Diagram 10 Diagram FHD

|  |  |  |
| --- | --- | --- |
|  |  | Wynikające z tego wymagania do interfejsu |
| Wiek | 6-18 | Blokowy |
| Wykształcenie i inteligencja | Podstawowe | Czytelny |
| Zdolności językowe | Język polski | W języku polskim |
| Zdolności manualne | Widzący, sprawny manualnie | Maksymalna prostota |
| Częstość użycia | Bardzo często | Standardowość |
| Wiedza o zadaniach | Niska | Blokowy |
| Obycie komputerowe | Użycie myszy, słabe zdolności pisania na klawiaturze | Prosta obsługa, mało lub brak pisania |

**Wymagania wobec interfejsu:**

Przydatność – aplikacja ma na celu zaznajomienie młodych ludzi z podstawami programowania, co jest przydatne dla użytkowników

Użyteczność – aplikacja ma być przeznaczona dla osób niezaznajomionych z komputerem więc musi być łatwa w użyciu

Łatwość opanowania – jako aplikacja edukacyjna, korzystanie z niej powinno być intuicyjne, możliwe do opanowania w krótkim czasie

Estetyka – aplikacja ma trafiać w gusta estetyczne osób młodszych, być wizualnie zachęcająca dla dzieci, a jednocześnie czytelna

**Kryteria oceny:**

Przydatność – czy aplikacja uczy użytkowników podstaw programowania

- metryka: bierzemy 5 kandydatów na studentów: dwóch uczniów szkoły podstawowej, dwóch uczniów liceum , jeden uczeń technikum i korzystają ze systemu przez semestr.

- miara: ilość wykonanych zadań w rankingu

Użyteczność – czy studenci byli w stanie przynajmniej raz uruchomić schemat

- metryka: bierzemy grupę studentów

- miara: platforma nauczania odnotowuje sprawdzenie wyniku(TAK/NIE)

Łatwość opanowania – w jakim średnio czasie wykonywano jedno zadanie

- metryka: bierzemy grupę nowych studentów

- miara: średni czas wykonania od pierwszego zalogowania do wykonania pierwszego zadania(czas)

Estetyka – średnia ilość dokonanych dostosowań systemu

- metryka: bierzemy grupę studentów

- miara: ilość zmian w dostosuj program(ilość)

# Aspekt UX

## Typ interfejsu

Interfejs systemu należy do kategorii WIMP (windows, icons, menus, pointer), oznacza to, że środowisko graficzne przypomina to z którego korzystają systemy takie jak Windows i Linux oraz programy z rodziny Office. Wybór został dokonany, ponieważ jest to standardowe rozwiązanie na które decydują się środowiska do wytwarzania oprogramowania, jaki jest również nasz system (np. Visual Studio). Zaletą takiego rozwiązania nad interfejsami WWW jest możliwość skupienia użytkownika na systemie oraz nieodciąganie go od przydzielonych zadań. Środowisko tekstowe natomiast wykluczyło by niezaznajomione osoby z technologią, takie jakie dotyczy nasz system.

## Wymagane urządzenia

Do obsługi naszego systemu niezbędna jest klawiatura oraz mysz.

## Grupy użytkowników

Aktorzy: Student

* Studenci:

Wiek: 8-18 lat,

Wykształcenie podstawowe lub w trakcie

Obycie z komputerem: słabe

Znajomość tematyki SI: słaba

Język: j. polski B1, j. angielski A1

## Typowe zadania

1-student:

Student (imię: Władysław, nazwisko: Jagiełło, login: wladyslaw.jagiello1) wybiera zadanie (nazwa: dodawanie zmiennych, id: 1). Wykonuje i uruchamia schemat(operacja dodawania, dwie zmienne niezmienne, console output).

## Scenariusze do zadania

Dla grupy studenci:

USTALENIA – Student jest zalogowany.

1. Student wybiera zadanie (nazwa: dodawanie zmiennych, id: 1)

Ekran wyboru zadań

1. System wyświetla początkowy schemat zadania

Ekran schematu z konsolą

1. Student wyświetla rozwijaną listę z dostępnymi blokami.

Ekran schematu z rozwijaną listą

1. Student łączy bloki

Ekran schematu łączenia bloków

1. System wyświetla wynik pracy schematu

Ekran schematu z konsolą

1. System wyświetla komunikat o poprawnym wykonaniu zadania

Ekran schematu z komunikatem o wykonaniu zadania

1. Student wraca do wyboru zadań

Ekran wyboru zadań

## Projekty Ekranów

### Szablon

**Chart

Description automatically generated**

Rysunek 5 Ekran Szablonu

### Ekran wyboru zadań

-Diagram

Description automatically generated with low confidence

Rysunek 6 Ekran wyboru- zadań

### Ekran schematu z konsolą

A picture containing box and whisker chart

Description automatically generated

Rysunek 7 Ekran schematu z konsolą

### Ekran schematu z rozwijaną listą

-Diagram

Description automatically generated

Rysunek 8 Ekran schematu z rozwijaną listą

### Ekran schematu łączenia bloków

Diagram

Description automatically generated

Rysunek 9 Ekran schematu łączenia bloków

### Ekran schematu z komunikatem o wykonaniu zadania

A picture containing diagram

Description automatically generated

Rysunek 10 Ekran schematu z komunikatem o wykonaniu zadania

# Testowanie interfejsu

Do testowania interfejsu zastosowaliśmy metodę z udziałem użytkownika polegającą na głośnym mówieniu. Do testów została wybrana 19 letnia osoba płci żeńskiej z niewielką wiedzą techniczną.

**Scenariusz  
1)** Zapoznanie użytkownika z pojęciem blokowych języków programowania za pośrednictwem systemu blueprint w Unreal Engine, komentarz użytkownika: „wyglądają jak połączone komórki w organizmie”, oraz przypomnienie definicji zmiennej.

**2 )** Użytkownikowi wyświetlony zostaje **ekran wyboru zadań** – użytkownik wybiera ikonkę domu, ponieważ „kursor miałam najbliższej tego”. Użytkownik sugerujemy wybór zadania – wybiera zadanie.

**4)** Użytkownikowi zostaje wyświetlony **ekran schematu z konsolą** – naciska przycisk znaku zapytania („nie wiedziałam co zrobić”), dowiaduje się, że pod prawym przyciskiem myszy kryje się ekran schematu z rozwijaną listą – zamyka okno pomocy.

**5)** Użytkownik podświetla kolejne (**w ekranie schematu z rozwijaną listą**) opcje z kolumny z pierwszej, aby następnie sprawdzać kolejne warstwy. Użytkownik wybiera jeden z bloków

**6)** Użytkownikowi ponownie **pojawia się ekran schematu** **z konsolą** zmodyfikowaną poprzez pojawienie się bloku. Użytkownik przesuwa blokiem po planszy oraz bawi się przybliżeniem i oddaleniem mapy. Użytkownik wyrzuca przypadkowo blok z polu widoku, nie domyśla się aby przesunąć widok w kierunku wyrzutu. Porównuje zachowanie interfejsu do działania programu „blender”.

**7)** Użytkownik naciska na wejścia bloku, naciska wyjście bloku i przytrzymuje(**ekran schematu łączenia bloków**) i próbuje połączyć wyjście z wejściem bloku. Nie wie co dalej, użytkownik otrzymuje propozycje dodania więcej bloków. Dodaje 3 kolejne bloki operacji dodawania (ekran schematu z rozwijaną listą), łączy bloki. Użytkownik otrzymuje poradę o dodanie zmiennych i outputu – użytkownik wykonuje propozycje. Prawidłowo wykonuje połączenia bloków, a następnie naciska przycisk „RUN”. Na konsoli pojawia się informacja, że „SI wykonało się” – użytkownik tego nie zauważa, zostaje poinstruowany o wyniku swojego działania. Użytkownik wprowadza inne wartości do bloków „5” i „3”, na konsoli pojawia się komunikat „SI wykonało się”.

**Wnioski**

Testowanie interfejsu ujawniło potrzebę zmiany domyślnego komunikatu w konsoli podczas ekranu ekran schematu z konsolą (poniekąd również ekranu schematu z rozwijaną listą i łączenia bloków) na „Tutaj pojawią się wyniki działania schematu, po naciśnięciu przycisku RUN”. Ponieważ wynik wykonania schematu często będzie wartością jedno cyfrową łatwą do przeoczenia.

Dodatkowo zorientowaliśmy, że nie zdefiniowaliśmy ekranu schematu z oknem pomocy.

**A picture containing diagram

Description automatically generated**Rysunek 11 Ekran schematu z konsolą poprawiony

Chart

Description automatically generated

*Rysunek 12 Ekran schematu z pomocą*

Kierując się strukturą WCAG 2.1 i zasadą postrzegalności proponujemy wprowadzenie udogodnienia w postaci rozbudowania toggle’a i oprócz możliwości przełączania pomiędzy motywem ciemnym i jasnym umożliwiał uruchomienie trybu wysokiego kontrastu. Chart

Description automatically generated with medium confidence

*Rysunek 13 Ekran szablonu z udogodnieniem dla niepełnosprawnych*

Graphical user interface, text, application

Description automatically generated

*Rysunek 14 Ekran szablonu z wysokim kontrastem*

*Graphical user interface

Description automatically generated*

*Rysunek 15 Ekran schematu z pomocą z wysokim kontrastem*

# Spis Diagramów

[Diagram 1 Biznesowy Diagram Przypadków Użycia 7](#_Toc121317897)

[Diagram 2 BPMN ucz się w programie 8](#_Toc121317898)

[Diagram 3 BPMN opis 9](#_Toc121317899)

[Diagram 4 BPMN Zarządzaj materiałami edukacyjnymi 10](#_Toc121317900)

[Diagram 5 Systemowy Diagram Przypadków Użycia 13](#_Toc121317901)

[Diagram 6 TODO: OPIS 15](#_Toc121317902)

[Diagram 7 Konceptualny Diagram Klas 22](#_Toc121317903)

[Diagram 8 Implementacyjny Diagram Klas 23](#_Toc121317904)

[Diagram 9 Diagram Obiektów 24](#_Toc121317905)

[Diagram 10 Diagram FHD 25](#_Toc121317906)

# Spis Ilustracji

[Rysunek 1 Formularz rejestracji 17](#_Toc121317946)

[Rysunek 2 Dyplom 18](#_Toc121317947)

[Rysunek 3 Oświadczenie RODO 19](#_Toc121317948)

[Rysunek 4 Faktura 20](#_Toc121317949)

[Rysunek 5 Ekran Szablonu 28](#_Toc121317950)

[Rysunek 6 Ekran wyboru zadań 29](#_Toc121317951)

[Rysunek 7 Ekran schematu z konsolą 29](#_Toc121317952)

[Rysunek 8 Ekran schematu z rozwijaną listą 30](#_Toc121317953)

[Rysunek 9 Ekran schematu łączenia bloków 30](#_Toc121317954)

[Rysunek 10 Ekran schematu z komunikatem o wykonaniu zadania 31](#_Toc121317955)