Dokumentacja projektu zaliczeniowego

Inżynierski Projekt Zespołowy

Autorzy: Mateusz Pawlikowski , Aleksander Grieger, Michał Maziarz, Piotr Wróbel

[Wprowadzenie: 5](#_Toc62374293)

[Ankieta 6](#_Toc62374294)

[Struktura pytań w ankiecie 6](#_Toc62374295)

[Pytania określające cechy osoby 6](#_Toc62374296)

[Pytania dotyczące trybu stacjonarnego i zdalnego 6](#_Toc62374297)

[Osobowości 7](#_Toc62374298)

[Formatowanie Excela 9](#_Toc62374299)

[Konstrukcja modelu 11](#_Toc62374300)

[Implementacja modelu w języku python: 14](#_Toc62374301)

[Funkcja licząca prawdopodobieństwo na podstawie wyników z ankiet 21](#_Toc62374302)

[Pobieranie wyników z ankiet oraz wyliczanie prawdopodobieństwa 21](#_Toc62374303)

[Losowanie odpowiedzi na podstawie prawdopodobieństwa 21](#_Toc62374304)

[Interfejs użytkownika 22](#_Toc62374305)

[Technologia i narzędzia 22](#_Toc62374306)

[Język programowania 22](#_Toc62374307)

[Zintegrowane środowisko programistyczne 22](#_Toc62374308)

[Narzędzia do stworzenia interfejsu 22](#_Toc62374309)

[Architektura 22](#_Toc62374310)

[Model 23](#_Toc62374311)

[Widok modelu 23](#_Toc62374312)

[Widok 23](#_Toc62374313)

[Wygląd intefejsu 23](#_Toc62374314)

[Główne okno 23](#_Toc62374315)

[Menu główne 24](#_Toc62374316)

[Sekcja File 24](#_Toc62374317)

[Sekcja Simulation 24](#_Toc62374318)

[Menu tworzenia miejsca pracy 24](#_Toc62374319)

[Drzewo projektu 25](#_Toc62374320)

[Panel główny 26](#_Toc62374321)

[Menu miejsca pracy 26](#_Toc62374322)

[Lista zespołów 26](#_Toc62374323)

[Zespół 27](#_Toc62374324)

[Członek zespołu 27](#_Toc62374325)

[Integracja interfejsu z symulacją 28](#_Toc62374326)

[Simulation Director 28](#_Toc62374327)

[Simulation Engine 30](#_Toc62374328)

[Przesyłanie danych do symulacji 30](#_Toc62374329)

[Integracja z językiem Python 31](#_Toc62374330)

[Interpreter języka Python 32](#_Toc62374331)

[Asynchroniczne wywołanie symulacji 32](#_Toc62374332)

[Klasy i ich relacje 32](#_Toc62374333)

[Model 32](#_Toc62374334)

[Interfejs IProjectItem 32](#_Toc62374335)

[Interfejs ITeamMember 33](#_Toc62374336)

[Intefejs ITeam 34](#_Toc62374337)

[Intefejs IWorkplace 36](#_Toc62374338)

[Interfejs IExperience 37](#_Toc62374339)

[Klasa Student 38](#_Toc62374340)

[Klasa Worker 38](#_Toc62374341)

[Klasa StudentExperience 38](#_Toc62374342)

[Klasa WorkerExperience 39](#_Toc62374343)

[Klasa Univercity 40](#_Toc62374344)

[Klasa Company 40](#_Toc62374345)

[Klasa Project 40](#_Toc62374346)

[Klasa TeamworkSimulationManager 41](#_Toc62374347)

[Interfejsy serwisów 42](#_Toc62374348)

[Klasa SimulationModel<T> 43](#_Toc62374349)

[Klasy TeamMemberSimulationModel 44](#_Toc62374350)

[Klasy WorkplaceSimulationModel 45](#_Toc62374351)

[Widok Modelu (View Model) 45](#_Toc62374352)

[Klasa NotifyPropertyChanges 45](#_Toc62374353)

[Klasa MainViewModel 45](#_Toc62374354)

[Klasa ProjectItemViewModel 46](#_Toc62374355)

[Klasa ProjectViewModel 46](#_Toc62374356)

[Klasa WorkplaceTemplateViewModel 47](#_Toc62374357)

[Klasa WorkplaceViewModel 47](#_Toc62374358)

[Klasa TeamViewModel 48](#_Toc62374359)

[Klasa TeamMemberViewModel 48](#_Toc62374360)

[Klasa StudentViewModel 49](#_Toc62374361)

[Klasa WorkerViewModel 49](#_Toc62374362)

[Klasa ExperienceViewModel 49](#_Toc62374363)

[Klasa StudentExperienceViewModel 49](#_Toc62374364)

[Klasa WorkerExperienceViewModel 50](#_Toc62374365)

[Klasa TeamMemberViewModelFactory 50](#_Toc62374366)

[Klasa ExperienceViewModelFactory 50](#_Toc62374367)

[Klasa ColorViewModel. 50](#_Toc62374368)

# 

# Wprowadzenie:

Celem projektu było opracowanie wieloagentowego symulatora umożliwiającego prowadzenie badań w zakresie porównania pracy zdalnej i stacjonarnej w zespołach pracowników firm około informatycznych i w zespołach studentów. Aby osiągnąć cel projektu opracowano model pracy zespołowej i zaimplementowano system informatyczny składający się z jądra symulacyjnego oraz interfejsu graficznego. Na uwagę zasługuje fakt integracji różnych technologii informatycznych wykorzystanych do implementacji systemu. Opracowany model zaimplementowano w języku Python a interfejs oraz komunikację z modelem w języku C#.

Aby rzetelnie odwzorować rzeczywistość skonstruowano ankietę w Google Docs. Ankieta została rozesłana do kilku firm informatycznych w szczecinie jaki i studentów różnych uczelni w Polsce.

W trakcie realizacji projektu opracowano następujące moduły:

-Ankieta walidacyjna

-Arkusz z odpowiedziami ankiety walidacyjnej

-Model pracy zespołowej pisany w języku python

-interfejs modelu pisany w języku C#

**Podział zadań:**

* Opracowanie założeń i konstrukcji modelu: Mateusz Pawlikowski, Piotr Wróbel, Michał Maziarz, Aleksander Grieger
* Konstrukcja ankiety dla studentów i pracowników: Mateusz Pawlikowski, Piotr Wróbel, Michał Maziarz, Aleksander Grieger
* Obróbka danych z ankiety w excelu –Aleksander Grieger
* System wyznaczenia osobowości z ankiety - Aleksander Grieger
* Algorytmy wyznaczania prawdopodobieństwa danego współczynnika w modelu w języku python: Michał Maziarz
* Interfejs aplikacji w C#: Mateusz Pawlikowski
* Integracja języka C# z pythonem: Mateusz Pawlikowski
* Komunikacja modelu z intefejsem :Mateusz Pawlikowski, Piotr Wróbel
* Implementacja modelu i jego logiki w języku python: Piotr Wróbel
* Dokumentacja : Mateusz Pawilkowski, Piotr Wróbel, Michał Maziarz, Aleksander Grieger

# Ankieta

## Struktura pytań w ankiecie

### Pytania określające cechy osoby

Tutaj znajdują się wszystkie pytania ,które mają na celu określić cechy osoby ankietowanej .Dla pracownika cechami są płeć ,wiek ,dziedzina pracy ,stanowisko pracy, czas pracy w aktualnej firmie i w aktualnej branży oraz osobowość , zaś dla studenta to płeć, kierunek ,rok ,stopień i tryb studiów(stacjonarny czy niestacjonarny) oraz osobowość.

.

### Pytania dotyczące trybu stacjonarnego i zdalnego

Reszta ankiety podzielona jest na zbiór pytań dotyczących kolejno trybu

stacjonarnego i zdalnego. Pytania zostały podzielone na pytania określające parametry aspektów pracy zespołowej, które są oceniane w skali od 1 do 5, oraz pytania kontrolne ,które zostały wprowadzone w celu sprawdzenia staranności wypełnienia ankiety.

# Osobowości

Wyróżniane są 4 główne typy osobowości:

1. Melancholik  
 2. Flegmatyk  
 3. Choleryk  
 4. Sangwinik

Oczywistym jest to, że nie wszyscy są zdefiniowani jako jeden typ i każdy może mieć coś w sobie z każdego typu, ale u każdego któryś z owych typów jest wiodący. Przykłady w jaki sposób łączą się dane typy:

1. Naturalne – sangwinik + choleryk / melancholik + flegmatyk  
 2. Komplementarne – flegmatyk + sangwinik / choleryk + melancholik  
 3. Przeciwstawne – sangwinik + melancholik / choleryk flegmatyk

Żeby stworzyć symulacje pracy złożonego przez nas zespołu, należało najpierw określić typy osobowości ankietowanych. W tym celu zebraliśmy 6 pytań pozwalających sklasyfikować daną osobę:

1. Wybierz słowo, które najlepiej Ciebie opisuje (1 - Zamknięty/a w sobie, 2 - Powściągliwy/a, 3 – Śmiały/a, 4 - Wylewny/a)

2. Wybierz słowo, które najlepiej Ciebie opisuje (1 - Wrażliwy/a, 2 - Nieszkodliwy/a, 3 -Przebiegły/a, 4 - Dowcipny/a)

3. Wybierz słowo, które najlepiej Ciebie opisuje (1 - Uzdolniony/a artystycznie, 2 -Zrównoważony/a, 3 - Odważny/a, 4 -Rozmowny/a)

4. Wybierz słowo, które najlepiej Ciebie opisuje (1 - Biorący/a wszystko do siebie, 2 - Cierpliwy/a, 3 - Pochopny/a, 4 -Entuzjastyczny/a)

5. Jestem (1 - Wierny/a, 2 - Opanowany/a, 3 - Zdecydowany/a, 4 Kontaktowy/a)

6. Jestem (1 - Nostalgiczny, 2 - Dystansujący/a się, 3 - Niecierpliwy/a, 4 - Roztargniony/a)

Odpowiedzi są tak skonstruowane, że kolejno odpowiadają danym typom.   
(1. Melancholik, 2. Flegmatyk, 3. Choleryk, 4. Sangwinik)

Jeśli ankietowany na większość pytań odpowie na jeden dany numer przypisany do danej osobowości to jest nim określany. (6 pytań po 4 odpowiedzi (typy))

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Warianty / Osobowości | Jedne z 4 typów | Jedne z 4 typów | Jedne z 4 typów | Jedne z 4 typów |
| Na 6 pytań odpowiada 6 razy na jeden typ | 6 | 0 | 0 | 0 |
| Na 6 pytań odpowiada 5 razy na jeden typ | 5 | 1 | 0 | 0 |
| Na 6 pytań odpowiada 4 razy na jeden typ | 4 | 1 | 1 | 0 |
| Na 6 pytań odpowiada 3 razy na jeden typ | 3 | 2 | 1 | 0 |
| Na 6 pytań odpowiada 3 razy na jeden typ | 3 | 1 | 1 | 1 |

We wszystkich tych wariantach wygrywa typ z większą ilością odpowiedzi oddanych.

W przypadku „remisu” nadal można określić typ osobowości u ankietowanego, ponieważ jak już było wspomniane są typy wiodące. Są również pytania i odpowiedzi bardziej zdecydowanie klasyfikujące.  
 (1. Melancholik; 2. Flegmatyk; 3. Choleryk; 4. Sangwinik)

Remis między 2 typami osobowości, czyli 3 odpowiedzi na jeden oraz 3 na drugi, po 0 na pozostałe trzeci i czwarty.  
 Na 6 pytań:

|  |  |  |
| --- | --- | --- |
| 3 razy odpowiedź nr: | 3 razy odpowiedź nr: | Wygrywa |
| 1. Melancholik | 2. Flegmatyk | 1. Melancholik |
| 1. Melancholik | 3. Choleryk | 3. Choleryk |
| 1 Melancholik | 4. Sangwinik | 4. Sangwinik |
| 2. Flegmatyk | 4. Sangwinik | 2. Flegmatyk |
| 2. Flegmatyk | 3. Choleryk | 3. Choleryk |
| 3. Choleryk | 4. Sangwinik | 4. Sangwinik |

Gdy zachodzi remis między 3 typami osobowości, czyli 2 odpowiedzi na jeden, 2 na drugi oraz 2 odpowiedzi na trzeci, po 0 na pozostały czwarty. Lub 2 odpowiedzi na jeden, 2 na drugi i po 1 odpowiedzi na trzeci oraz czwarty. W tym przypadku decydująca jest odpowiedź na pytanie numer 3

(3. Wybierz słowo, które najlepiej Ciebie opisuje (1 - Uzdolniony/a artystycznie, 2 -Zrównoważony/a, 3 - Odważny/a, 4 -Rozmowny/a), ponieważ ono najlepiej ukazuje typ ankietowanego i z czym się najbardziej utożsamia.

# Formatowanie Excela

Aby ułatwić tworzenie modelu w pythonie, należało sformatować odpowiedzi z ankiety w excelu. Polegało to na tym, że już na poziomie excela był określany typ osobowości ankietowanego:  
 1. Należało zliczyć u każdego ilość odpowiedzi na dany typ.  
 2. Sprawdzić jakie zależności zachodzą:   
- 4 odpowiedzi lub więcej razy jeden typ  
 - 3 odpowiedzi na jeden typ, bez remisu z innym (3-2-1-0 / 3-1-1-1)  
 - 3 odpowiedzi na jeden typ oraz 3 odpowiedzi na drugi, czyli remis i wyłonienie wiodącego  
 - 2 odpowiedzi na jeden typ oraz 2 odpowiedzi na drugi, po 1 na pozostałe, czyli remis i wyłonienie wiodącego  
 - po 2 odpowiedzi na 3 typy, czyli remis i wyłonienie wiodącego  
 3. Przepisanie do osobnej kolumny określonych już typów osobowości

Należało również zmienić na poziomie excela, odpowiedzi tekstowe na liczbowe odpowiedniki dla ułatwienia sczytywania w pythonie i pisania modelu:

Zmiana nastąpiła w następujących pytaniach:

ANKIETA DLA PRACOWNIKÓW:

**1. Jaka jest Twoja płeć?**  
Odpowiedzi: “Mężczyzna” = 1; “Kobieta” = 2;

**2. Zaznacz swój przedział wiekowy.**  
Odpowiedzi: "20-30" = 1; "30-40" = 2; "40-50" = 3; “50+” = 4;

**3. Jaka jest dziedzina Twojej pracy?**  
Odpowiedzi: "Branża IT" = 1; "Badania i rozwój" = 2; "Administracja" = 3; "Edukacja" = 4; ”Inne” = 5;

**4. Jakie jest Twoje stanowisko w pracy?**  
Odpowiedzi: "Dyrektor/Prezes" = 1; "Kierownik/lider zespołu" = 2; "Starszy specjalista" = 3; "Specjalista" = 4;   
"Młodszy specjalista" = 5; "Praktykant/Stażysta" = 6; ”Inne” = 7;

**5. Jak długo pracujesz w aktualnej firmie?**  
Odpowiedzi: "Mniej niż rok" = 1; "Od roku do dwóch lat" = 2; "Około 5 lat" = 3; “Więcej niż 5 lat“ = 4;

**6. Jak długo pracujesz w aktualnej branży?**  
Odpowiedzi: "Mniej niż rok" = 1; "Od roku do dwóch lat" = 2; "od 3 do 5 lat" = 3; "od 6 do 10 lat" = 4;  
 “powyżej 10 lat” = 5;

**7. Czy preferujesz pracę stacjonarną?**  
Odpowiedzi: “Tak” = 1; “Nie” = 2; “Nie ma to dla mnie znaczenia” = 3;

**8. Czy preferujesz pracę zdalną?**  
Odpowiedzi: “Tak” = 1; “Nie” = 2; “Nie ma to dla mnie znaczenia” = 3;

**9. Z jakiego komunikatora korzystasz?**  
Odpowiedzi: "Microsoft teams" = 1; “Zoom" = 2; "Slack" = 3; "Discord" = 4; ”Inne” = 5;

**10. Czy korzystasz z kamery video podczas spotkań w pracy zdalnej**?  
 Odpowiedzi: “Tak” = 1; “Nie” = 2;

ANKIETA DLA STUDENTÓW:

**1. Jaka jest Twoja płeć?**  
Odpowiedzi: “Mężczyzna” = 1; “Kobieta” = 2;

**2. Jaki kierunek studiujesz?**  
Odpowiedzi: "Informatyka" = 1; "Ekonomia" = 2; "Zarządzanie" = 3; "Psychologia" = 4; "Mechatronika" = 5; "Medycyna" = 6; “Inne” = 7;

**3.Wybierz stopień studiów?**  
 **Odpowiedzi: "Licencjat" = 1; "Inżynier" = 2; "Magister" = 3; "Doktorat" = 4;**

**4. Studiujesz stacjonarnie czy niestacjonarnie?**  
Odpowiedzi: “Stacjonarne” = 1; “Niestacjonarne” = 2;

**5. Czy preferujesz naukę stacjonarną?**  
Odpowiedzi: “Tak” = 1; “Nie” = 2; “Nie ma to dla mnie znaczenia” = 3;

**6. Ile minimum masz czasu na przerwę podczas nauki stacjonarnej (w minutach)?**  
Odpowiedzi: “0” = 1; “15” = 2; "30” = 3; “60” = 4; "120” = 5; “150” = 6;

**7. Czy wykonujesz dodatkowe aktywności związane z funkcjonowaniem uczelni w trybie stacjonarnym?**  
Odpowiedzi: “Tak” = 1; “Nie” = 2;

**8. Czy preferujesz naukę zdalną?**  
Odpowiedzi: “Tak” = 1; “Nie” = 2; “Nie ma to dla mnie znaczenia” = 3;

**9. Ile minimum masz czasu na przerwę podczas nauki zdalnej (w minutach)?**  
Odpowiedzi: “0” = 1; “15” = 2; "30” = 3; “60” = 4; "120” = 5; “150” = 6;

**10.Czy wykonujesz dodatkowe aktywności związane z funkcjonowaniem uczelni w trybie stacjonarnym?**  
Odpowiedzi: “Tak” = 1; “Nie” = 2;

**11. Z jakiego komunikatora korzystasz?**  
Odpowiedzi: "Microsoft teams" = 1; “Zoom" = 2; "Slack" = 3; "Discord" = 4; ”Inne” = 5;

**12. Czy korzystasz z kamery video podczas spotkań w pracy zdalnej**?  
 Odpowiedzi: “Tak” = 1; “Nie” = 2;

# Konstrukcja modelu

Model został podzielony na 3 aspekty pracy zespołowej w każdej z nich znajduję się kilka parametrów odpowiadających danej kategorii.

**Symulacja pracy zespołowej**

3 aspekty (przed COVID, po COVID):

1. Samopoczucie członków zespołu klasa: (Comfort)
2. Zmęczenie po pracy (tygodniowo)(dwa pytania podobnej treści drugie 1h)
3. Opinia o trybie pracy
4. Liczba/długość/jakość przerw
5. atmosfera pracy (socjologia w pracy)

e. Ergonomia pracy

f. Chęć samodoskonalenia uczenia zwiększania swoich kompetencji

g. Zmęczenie po pracy (dzienne)

1. Efektywność zespołu

a. Realizacja celów założeń projektu

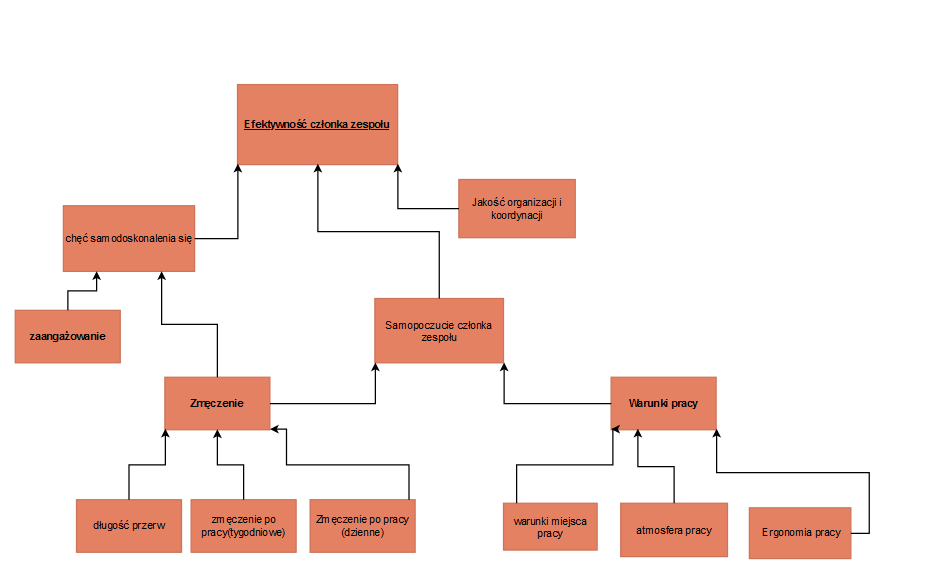
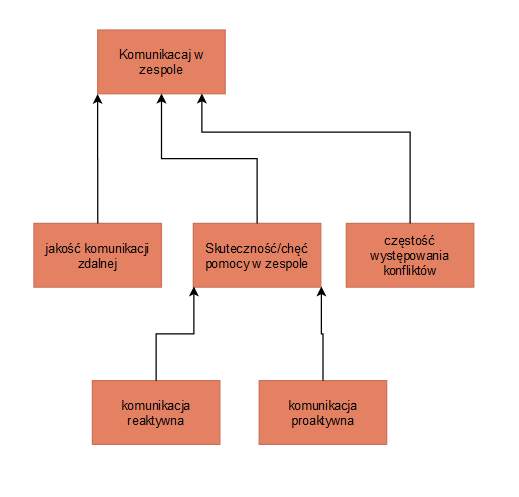
b.opóźnienia

c. Zaangażowanie

d. Jakość organizacji i koordynacji

1. Komunikacja w zespole
2. Skuteczność/chęć pomocy w zespole
3. Komunikacja pro-aktywna
4. Komunikacja reaktywna
5. Jakość komunikacji zdalnej (komunikatory teams Itp.) (dodatkowy parametr przy pracy zdalnej)
6. Skuteczność/jakość komunikatora
7. Wideo + Audio
8. Rozumienie wypowiedzi i kontekstu w aspekcie zdalnym/stacjonarnym
9. Częstość występowania konfliktów

Poniższy graf prezentuje zależności poszczególnych parametrów z których korzysta model:



Model na podstawie poszczególnych cech członków zespołu losuje wartości współczynników (od 1 do 5) z prawdopodobieństwem wyznaczonym na podstawie wyników ankiet według określonego wzoru:

Losowanie wartości z prawdopodobieństwa pod względem danej cechy lub n cech:

P.wyboru1=P( odp1( C1 ) / l.os.odp( C1 ) ) + P( odp1( C2 ) / l.os.odp( C2 ) )+ ... +P(odp1( Cn ) / l.os.odp( Cn)

P.wyboru2=P( odp2( C1 ) / l.os.odp( C1 ) ) + P( odp2( C2 ) / l.os.odp( C2 ) )+ ... +P(odp2( Cn ) / l.os.odp( Cn)

.

.

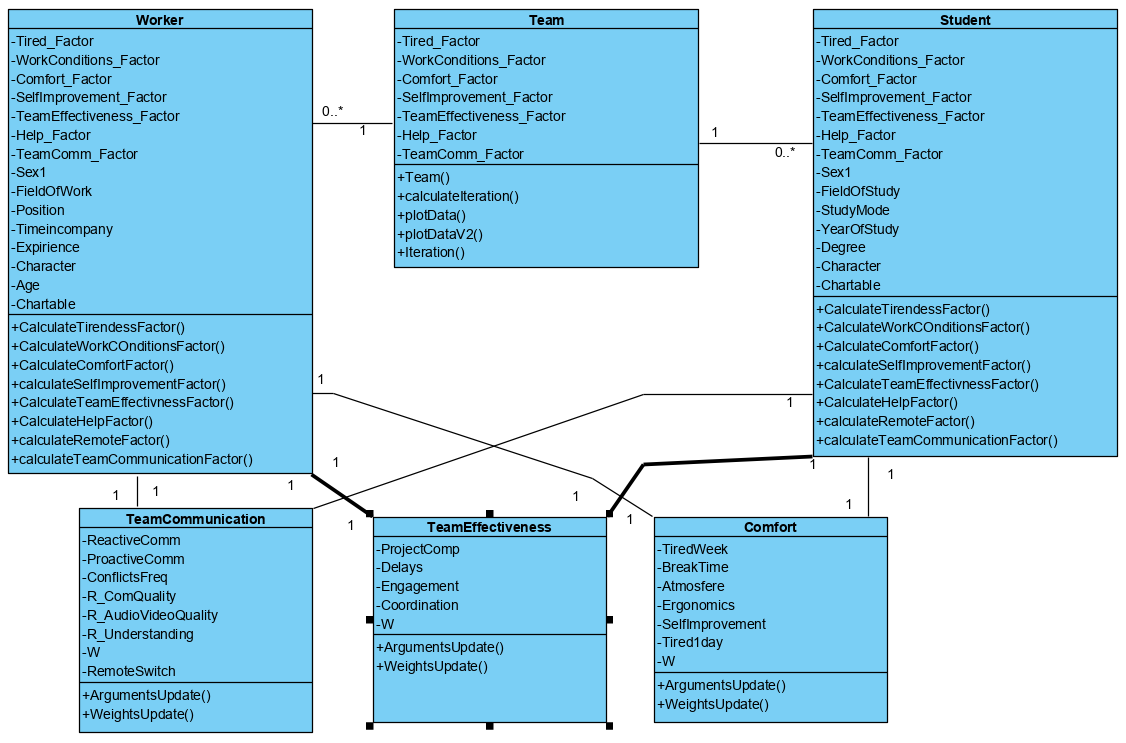
.

P.wyboru5=P( odp5( C1 ) / l.os.odp( C1 ) ) + P( odp5( C2 ) / l.os.odp( C2 ) )+ ... +P(odp5( Cn ) / l.os.odp( Cn)

Następnie wartość tych współczynników używane są do wyznaczenia poszczególnych cech zespołu przy czym każda z cech posiada własny współczynnik wagi który pozwala dostrajać model.

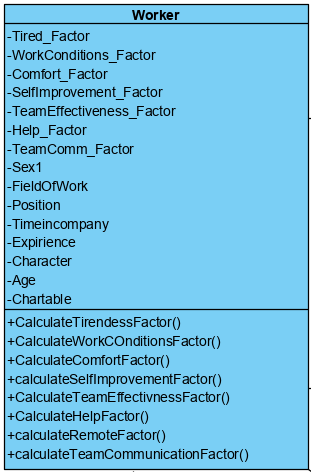
# Implementacja modelu w języku python:

Diagram klas:



Klasa Team posiada listę klas Student bądź Worker w zależności od rodzaju zespołu. Klasy TeamCommunication, TeamEffectivness, Comfort są w relacji jeden do jednego z klasą Worker i Student, przechowywują parametry poszczególnych aspektów pracy zespołowej w iteracjach.

Klasa **Worker**



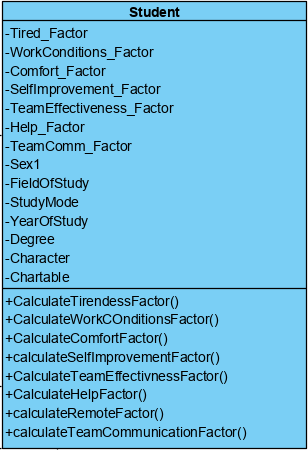
Elementy składowe klasy **Worker** to:

* Pole **Tired\_Factor** -lista przechowuje wartość współczynnika zmęcznenia
* Pole **WorkConditions\_Factor** - lista przechowuje wartość współczynnika warunków pracy
* Pole **Comfort\_Factor** - lista przechowuje wartość współczynnika komfortu pracy
* Pole **SelfImprovement\_Factor** - lista przechowuje wartość współczynnika samodoskolnalenia się
* Pole **TeamEffectivness** – lista przechowuje wartość współczynnika efektywności zespołu
* Pole **Help\_Factor** - lista przechowuje wartość współczynnika pomocy
* Pole **TeamComm\_Factor –** lista przechowuje wartość współczynnika komunikacji w zespole
* Pole **Sex** - przechowuje dane o płci pracownika
* Pole **FieldOfWork** - przechowuje dane o dziedzinie pracy pracownika
* Pole **Position** - przechowuje dane o stanowisku pracownika(senior junior itp)
* Pole **Timeincompany** - przechowuje dane o stażu pracownika w danej firmie
* Pole **Position** - przechowuje dane o stanowisku pracownika(senior junior itp)
* Pole **Expirience** – przechowuje dane o doświadczeniu zawodowym pracowika
* Pole **Character** przechowuje informacje o typie osobowości pracownika
* Pole **Chartable** – przechowuje wartości powyższych pól klasy

Metody:

Analogicznie każda z metod służy do obliczenia średniej ważonej z powyższych współczynników.

Klasa **Student**



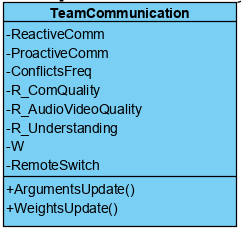
Elementy składowe klasy Studentto:

* Pole **Tired\_Factor** - lista przechowuje wartość współczynnika zmęcznenia
* Pole **WorkConditions\_Factor** - lista przechowuje wartość współczynnika warunków pracy
* Pole **Comfort\_Factor** - lista przechowuje wartość współczynnika komfortu pracy
* Pole **SelfImprovement\_Factor** - lista przechowuje wartość współczynnika samodoskolnalenia się
* Pole **TeamEffectivness\_Factor** – lista przechowuje wartość współczynnika efektywności zespołu
* Pole **Help\_Factor** - lista przechowuje wartość współczynnika pomocy
* Pole **TeamComm\_Factor** - lista przechowuje wartość współczynnika komunikacji w zespole
* Pole **Sex** - przechowuje dane o płci pracownika
* Pole **FieldOfStudy** - przechowuje dane o kierunku studiowanym przez studenta
* Pole **StudyMode** - przechowuje dane o trybie studiowania studenta
* Pole **YearOfStudy** - przechowuje dane o orku studiów studenta
* Pole **Degree** - przechowuje dane stopniu studiów
* Pole **Character** przechowuje informacje o typie osobowości studenta
* Pole **Chartable** – przechowuje wartości powyższych pól klasy

Metody:

Analogicznie każda z metod służy do obliczenia średniej ważonej z powyższych współczynników.

Klasa **TeamCommunication**



Elementy składowe klasy TeamCommunicationto:

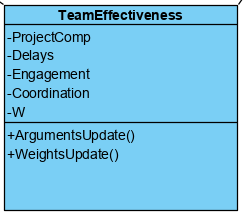
* Pole **ReactiveComm** przechowywujace wartość współczynnika komunikacji reaktywnej
* Pole **ProactiveComm** przechowywujace wartość współczynnika komunikacji proaktywnej
* Pole **R\_ComQuality** przechowywujace wartość współczynnika jakości komunikacji zdalnej
* Pole **R\_AudioVideoQuality** przechowywujace wartość współczynnika jakości audio i ivdeo podczas komunikacji zdalnej
* Pole **R\_Understanding** przechowywujace wartość współczynnika rozumienia kontekstu wypowiedzi
* Pole **W** przechowywuje wagi do powyższych współczyników na podstawie których wyznaczana jest średnia ważona

Metody:

**ArgumentsUpdate** – metoda aktualizuje wartości współczynników podczas iteracji

**WeightsUpdate** - Metoda aktualizuje wartości wag współczyników podczas iteracji

Klasa **TeamEffectiveness**



Elementy składowe klasy **TeamCommunication** to:

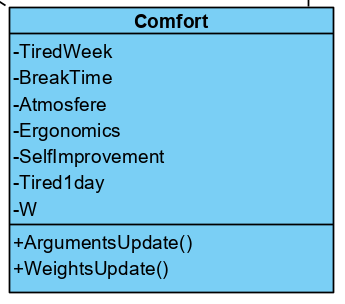
* Pole **ProjectComp** przechowywujace wartości współczynnika realizacji celów i założeń
* Pole **Delays** przechowywujace wartość współczynnika opóźnień w projekcie
* Pole **Engagement** przechowywujace wartość współczynnika jakości komunikacji zdalnej
* Pole **Coordinatio**n przechowywujace wartość współczynnika jakości audio i video podczas komunikacji zdalnej
* Pole **W** przechowywuje wagi do powyższych współczynników na podstawie których wyznaczana jest średnia ważona

Metody:

**ArgumentsUpdate** – metoda aktualizuje wartości współczynników podczas iteracji

**WeightsUpdate** - Metoda aktualizuje wartości wag współczynników podczas iteracji

Klasa **Comfort**



Elementy składowe klasy **Comfort** to:

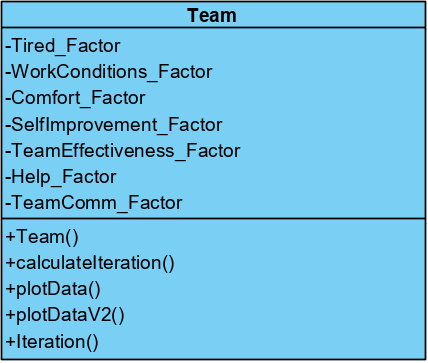
* Pole **TiredWeek** przechowywujace wartości współczynnika zmęczenia po tygodniu pracy
* Pole **BreakTime** przechowywujace wartość współczynnika ilości przerw w trakcie pracy
* Pole **Atmosfere** przechowywujace wartość współczynnika atmosfery pracy
* Pole **Ergonomics** przechowywujace wartość współczynnika ergonomii stanowiska pracy
* Pole **SelfImprovement** przechowywujace wartość współczynnika samodoskonalenia się
* Pole **Tired1day** współczynnika zmęczenia po dniu pracy
* Pole **W** przechowywuje wagi do powyższych współczynników na podstawie których wyznaczana jest średnia ważona

Metody:

**ArgumentsUpdate** – metoda aktualizuje wartości współczynników podczas iteracji

**WeightsUpdate** - Metoda aktualizuje wartości wag współczynników podczas iteracji

Klasa **Team**



Elementy składowe klasy **Comfort** to:

* Pole **Tired\_Factor** - lista przechowywuje wartość współczynnika zmęczenia
* Pole **WorkConditions\_Factor** - lista przechowywywuje wartość współczynnika warunków pracy
* Pole **Comfort\_Factor** - lista przechowywuje wartość współczynnika komfortu pracy
* Pole **SelfImprovement\_Factor** - lista przechowywuje wartość współczynnika samodoskolnalenia się
* Pole **TeamEffectivness\_Factor** – lista przechowywuje wartość współczynnika efektywności zespołu
* Pole **Help\_Factor** - lista przechowywuje wartość współczynnika pomocy
* Pole **TeamComm\_Factor** - lista przechowywuje wartość współczynnika komunikacji w zespole
* Team- konstruktor zespoły pozwalający stworzyć zespół studentów lub pracowników
* Metoda calculate iteration wylicza średnie ze współczynników zespołu i dodaje je do list
* Metoda plotData generuje wykresy współczynników
* Metoda plotDataV2 generuje wykresy współczynników w jednym oknie dialogowym
* Metoda iteration przeprowadza zadaną ilość iteracji

## Funkcja licząca prawdopodobieństwo na podstawie wyników z ankiet

Funkcja licząca prawdopodobieństwo złożona jest z dwóch części. Pobrania wyników z ankiet i wyliczenia prawdopodobieństwa oraz losowania odpowiedzi na podstawie prawdopodobieństwa.

### Pobieranie wyników z ankiet oraz wyliczanie prawdopodobieństwa

W zależności od wyboru typu członków zespołu ,pobierany zostanie excel dotyczący pracowników albo studentów. Na podstawie każdej z wcześniej określonych cech obliczane zostanie prawdopodobieństwo wybrania danej wartości parametrów ,a następnie uzyskane zostanie średnie prawdopodobieństwo ze wszystkich cech.

### Losowanie odpowiedzi na podstawie prawdopodobieństwa

Na podstawie uzyskanego prawdopodobieństwa dla wszystkich cech członka zespołu zostanie stworzona tabela zawierająca tyle wartości danego parametru ile wyniosło prawdopodobieństwo pomnożone przez 100.Na koniec z tabeli pobierana zostanie losowo wybrana wartość dla parametru.

# Interfejs użytkownika

Interfejs użytkownika został zaprojektowany aby umożliwić przygotowanie danych potrzebnych modelowi do wykonania symulacji pracy zespołowej oraz zapisanie tych danych.

Najważniejszym celem przy tworzeniu interfejsu była jego niezależność od modelu symulacji napisanego w języku Python, dzięki czemu interfejs może współpracować z modelem napisanym w całkowicie innym języku.

## Technologia i narzędzia

### Język programowania

Jako język programowania, w którym zaprogramowano intefejs użytkownika, użyto języka C# w wersji 7.0 na platformie .NET Core.

### Zintegrowane środowisko programistyczne

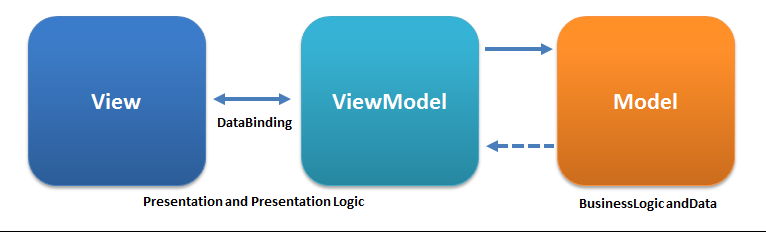
Głównym narzędziem do pracy nad interfejsem było zintegrowane środowisko programistyczne Visual Studio Preview firmy Microsoft.

#### Narzędzia do stworzenia interfejsu

Do utworzenia aplikacji desktopowej skorzystano z Windows Presentation Foundation (WPF).

## Architektura

Interfejs został zaprojektowany zgodnie z architekturą ModelView-ViewModel (MVVM).



Decyzja ta pozwala oddzielić dane związane z modelem od widoku dzięki czemu będzie można zaimplementować interfejs użytkownika w innej technologii lub dla innego systemu operacyjnego bez konieczności zmiany istniejącego już kodu modelu.

### Model

W modelu zawarte zostały klasy definiujące dane dla symulacji pracy zespołowej oraz logikę, która pozwala przekazać dane do symulacji i ją uruchomić.

### Widok modelu

Zgodnie z założeniami architektury dane, które zostały zdefiniowane w modelu mają swój widok-modelu, który uzupełnia je o informacje potrzebne do ich poprawnej reprezentacji w widoku.

### Widok

Widok interfejsu został zaprojektowany w narzędziu Windows Presentation Foundation (WPF) przy współpracy języka C# z językiem opisu interfejsu użytkownika Extensible Application Markup Language (XAML). Każde dane potrzebne dla symulacji pracy zespołowej mają swój widok.

## Wygląd interfejsu

### Główne okno

Wygląd głównego okna został zaprojektowany tak aby posiadał jak najwięcej potrzebnych kontrolek i wyświetlanych informacji do uruchomienia symulacji zachowując przy tym łatwość nawigacji po interfejsie.



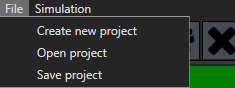
### Menu główne

W menu głównym są dwie sekcje: File oraz Simulation.



#### Sekcja File

Sekcja File zawiera opcje pozwalające na stworzenie projektu na nowo, zapisanie aktualnego projektu w pliku oraz załadowanie projektu już z istniejącego pliku.



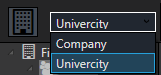
#### Sekcja Simulation

Sekcja Simulation pozwala na uruchomienie symulacji.



### Menu tworzenia miejsca pracy

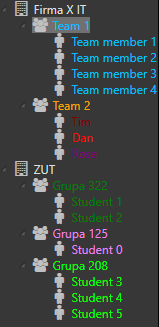
Do utworzenia nowego miejsca pracy danego typu istnieje specjalne menu, które składa się z przycisku dodającego miejsce pracy i z listy rozwijanej zawierającej typy miejsc pracy.



### Drzewo projektu

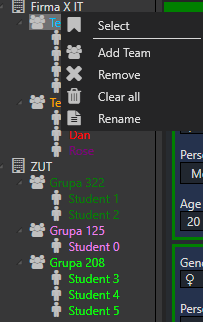
Aby ułatwić nawigację w projekcie, po lewej stronie interfejsu zostało umieszczone drzewo projektu, które przedstawia hierarchicznie wszystkie dodane miejsca pracy i drużyny zawarte w nich.

Każdy element drzewa aktualizuje się zgodnie z obiektem, z którym jest związany, np. zmiana koloru lub nazwy.



Drzewo pozwala również na łatwe wprowadzanie zmian w projekcie bez szukania odpowiedniego obiektu w głównym panelu z jego poziomu co ułatwia pracę. Te zmiany to:

* Zmiana nazwy
* Dodanie odpowiedniego elementu (jeśli to możliwe)
* Usunięcie elementu
* Usunięcie wszystkich elementów z “rodzica”
* Wybranie miejsca pracy do wyświetlenie w głównym panelu



Do odróżnienia elementów w drzewie są wyświetlane odpowiednie ikonki sugerujące jakiego typu jest dany element, np. element zespołu posiada ikonkę grupy.

### Panel główny

#### Menu miejsca pracy

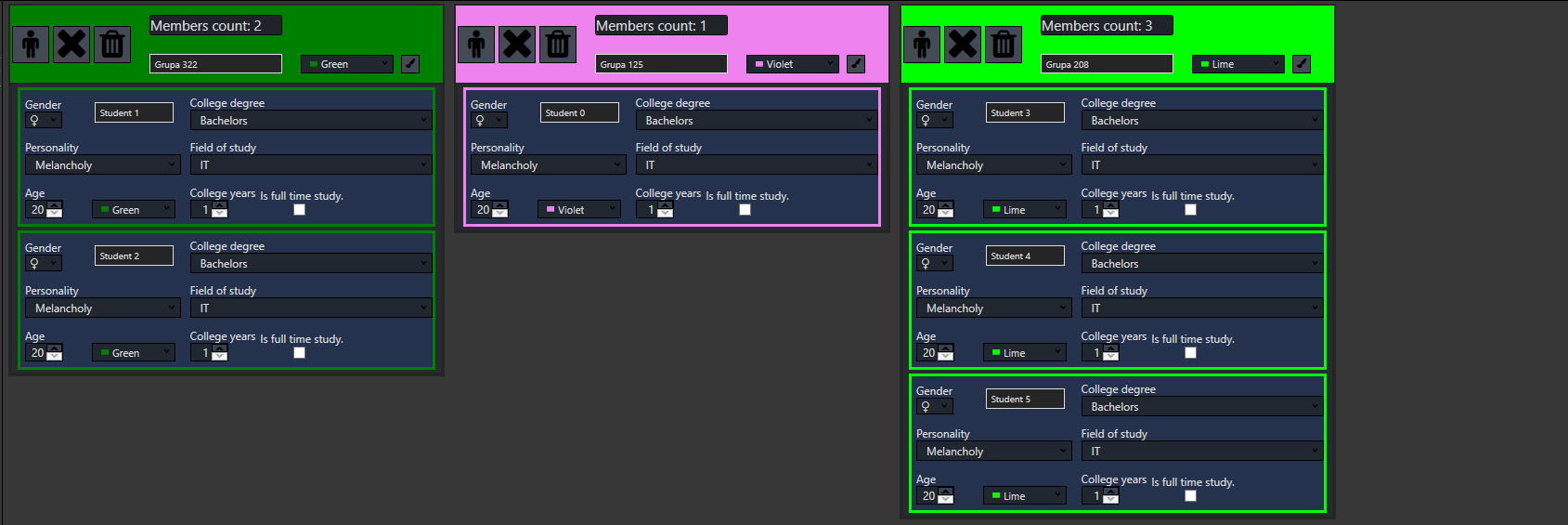
Do wprowadzania zmian w miejscu pracy służy menu, które zawiera:

* Przycisk do utworzenia nowego zespołu.
* Przycisk do usunięcia zaznaczonego zespołu.
* Przycisk do usunięcia wszystkich zespołów.
* Pole tekstowe do zmiany nazwy miejsca pracy.
* Informacja o liczbie zespołów.
* Check box do zaznaczenia czy miejsce pracy jest w trybie stacjonarnym czy zdalnym.



#### Lista zespołów

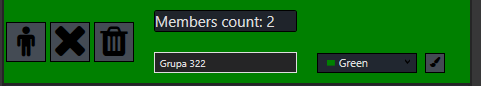
Każde miejsce pracy ma kontrolkę do wyświetlenia listy zespołów.



#### Zespół

Kontrolka zespołu służy do kontrolowania członków zespołu. Składa się z:

* Przycisku do dodania nowego członka zespołu.
* Przycisku do usunięcia zaznaczonego członka zespołu.
* Przycisku do usunięcia wszystkich członków zespołu.
* Informacji o liczbie członków zespołu
* Pola tekstowego pozwalającego na zmianę nazwy zespołu.
* Listy rozwijanej do wyboru koloru zespołu.
* Przycisku, który pozwala na pokolorowanie wszystkich członków zespołu wybranym kolorem z listy rozwijanej.

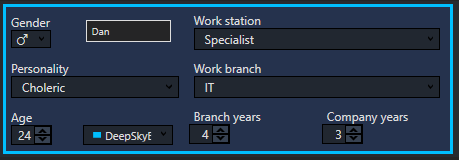
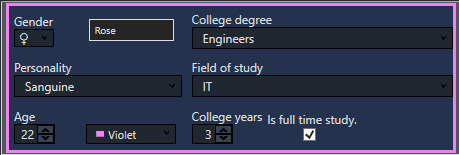


#### Członek zespołu

Kontrolka członku zespołu pozwala na zmianę danych danego członka zespołu. Kontrolka składa się z głównych elementów związanych z każdym członkiem zespołu, czyli:

* Lista rozwijana do wyboru płci.
* Lista rozwijana do wyboru osobowości.
* Pole liczbowe do wprowadzenia wieku.
* Lista rozwijana do wyboru koloru.
* Pole tekstowe do zmiany nazwy członka zespołu.

W modelu są brani pod uwagę różni członkowe zespołu, którzy wymagają wprowadzenia więcej informacji, dlatego kontrolka zawiera miejsce do umieszczenia dodatkowych elementów.

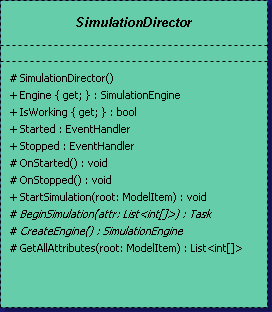
* Elementy dla członka zespołu “Worker” to:
  + Lista rozwijana stanowisk pracy.
  + Lista rozwijana branż pracy.
  + Pole liczbowe do wprowadzenia lat w branży.
  + Pole liczbowe do wprowadzenia lat w firmie.  
    
* Elementy dla członka zespołu “Student” to:
  + Lista rozwijana stopni studiów.
  + Lista rozwijana kierunków studiów.
  + Pole liczbowe lat na studiach.
  + Check box do zaznaczenia czy student studiuje w trybie stacjonarnym czy niestacjonarnym.  
    

## Integracja interfejsu z symulacją

Głównym celem przy tworzeniu interfejsu była niezależność tego interfejsu od symulacji, dlatego aby uruchamiać symulację stworzoną w dowolnej technologii zostały utworzone dwie abstrakcyjne klasy, tj. **SimulationDirector** oraz **SimulationEngine**

### Simulation Director

Abstrakcyjna klasa **SimulationDirector** służy do kontrolowania symulacji. W niej należy zaimplementować w jaki sposób uruchamiana jest symulacja. Do łatwiejszej pracy przy uruchamianiu symulacji została w klasie **SimulationDirector** zdefiniowana metoda wytwórcza, która zwraca odpowiedni typ klasy **SimulationEngine**, która to pozwala wykonywać program w innej technologii.

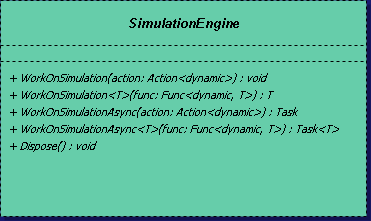


Elementy składowe klasy **SimulationDirector** to:

* Właściwość **Engine** – Zwraca obiekt klasy **SimulationEngine**, który został utworzony w metodzie wytwórczej **CreateEngine**.
* Właściwość **IsWorking** – Zwraca czy symulacja jest uruchomiona i się wykonuje. Używana tylko jeśli symulacja jest wykonywana asynchronicznie.
* Zdarzenie **Started** - Wywołuje się kiedy symulacja się rozpoczęła.
* Zdarzenie **Stopped** - Wywołuje się kiedy symulacja skończyła pracę.
* Metoda **OnStarted** – Wirtualna “protected” metoda, która pozwala klasom podrzędnym wywołanie zdarzenia **Started**.
* Metoda **OnStopped** – Wirtualna “protected” metoda, która pozwala klasom podrzędnym wywołanie zdarzenia **Stopped**.
* Metoda **StartSimulation** – Publiczna metoda, która uruchamia symulację przyjmując główny element projektu zawierający dane dla modelu (wyjaśnione dalej w podrozdziale “ “).
* Metoda **BeginSimulation** - “Protected” abstrakcyjna metoda uruchamiająca symulację asynchronicznie. Jako parametr przyjmuje listę tablic z odpowiednimi danymi dla symulacji.
* Metoda **CreateEngine** - “Protected” abstrakcyjna metoda wytwórcza do stworzenia odpowiedniego typu instancji klasy **SimulationEngine**.
* Metoda **GetAllAttributes** - “Protected” metoda, która zwraca listę tablic z odpowiednimi danymi dla symulacji przeszukując całe drzewo projektu zaczynając od danego elementu przesłanego przez parametr **root**.

### Simulation Engine

Abstrakcyjna klasa, która odpowiada za uruchamianie programu w innej technologii (w innym języku programowania). Implementuje interfejs **IDisposable**.



Elementy składowe klasy **SimulationEngine** to:

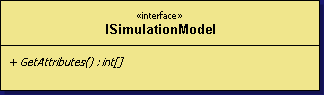
* Metoda **WorkOnSimulation(Action<dynamic> action)** - abstrakcyjna metoda, która uruchamia kod w innej technologii przesłany przez delegat **action**.
* Metoda **WorkOnSimulation<T>(Func<dynamic, T> func)** - abstrakcyjna metoda generyczna, która uruchamia kod w innej technologii przesłany przez delegat **func**, zwracając również dane o typie wskazanym przez parametr generyczny **T**.
* Metoda **WorkOnSimulationAsync(Action<dynamic> action)** - zwraca obiekt **Task**, który wywołuje metodę **WorkOnSimulation(Action<dynamic> action)**.
* Metoda **WorkOnSimulationAsync<T>(Func<dynamic, T> func)** - zwraca obiekt **Task**, który wywołuje metodę **WorkOnSimulation<T>(Func<dynamic, T> func)**.
* Metoda **Dispose()** - metoda z interfejsu **IDisposable**, czyści wszystkie nieobsługiwane przez **GC** obiekty.

### Przesyłanie danych do symulacji

Przy komunikacji interfejsu z symulacją istnieje założenie, że interfejs wysyła dane potrzebne symulacji w fromie tablic typu **Integer** w prawidłowej kolejności, natomiast tablice muszą zawierać prawidłowo dobrane liczby w odpowiednich indeksach.

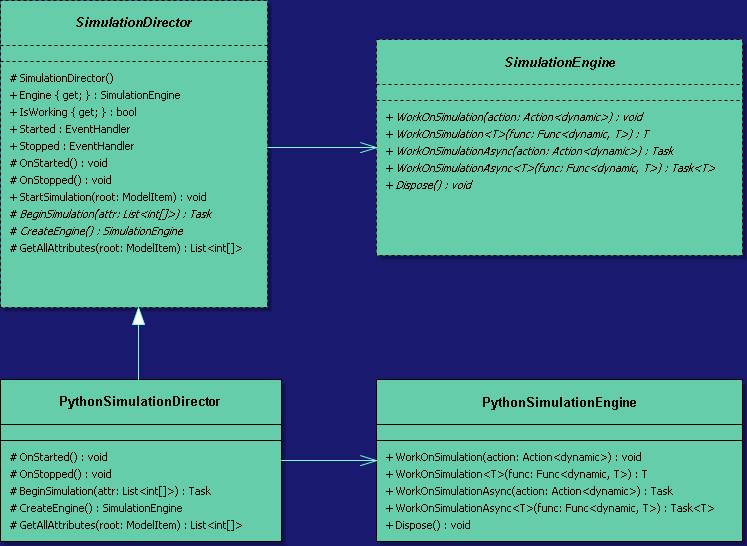
Każda klasa definiująca dane potrzebne symulacji dziedziczy po abstrakcyjnej klasi **ModelItem**. Klasa ta zawiera właściwość typu **ISimulationModel**, która definiuje metodę **GetAttributes** zwracającą odpowiednią tablicę dla danego typu klasy, która zostanie przesłana do symulacji.

Klasy dzedziczące po **ModelItem** muszą same zainicjalizować właściwość **SimulationModel** odpowiednim typem implementującym interfejs **ISimulationModel**.


### Integracja z językiem Python

Do integracji z symulacją napisaną w języku programowania Python zostały odpowiednio zaimplementowane klasy **SimulationDirector** oraz **SimulationEngine** pod nazwami **PythonSimulationDirector** i **PythonSimulationEngine**. Do wywoływania kodu z języka Python została wykorzystana biblioteka [Python.NET](http://pythonnet.github.io).



#### Interpreter języka Python

Interpreter języka Python wykorzystywany do uruchomienia symulacji jest w wersji 3.7.

#### Asynchroniczne wywołanie symulacji

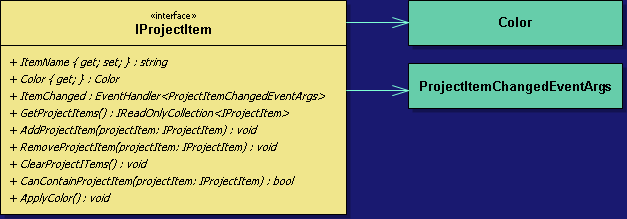
Na ten moment wywołanie asynchroniczne symulacji w języku Python jest niezalecane, ponieważ jedynie pierwsze uruchomienie symulacji się powiedzie.

## Klasy i ich relacje

### Model

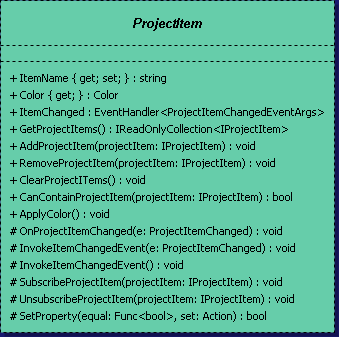
#### Interfejs IProjectItem

Każdy element, który jest częścią projektu wykorzystywanego przez symulację implementuje interfejs **IProjectItem**. Intefejs ten służy głównie do stworzenia drzewa projektu i łatwego go przeszukiwania.



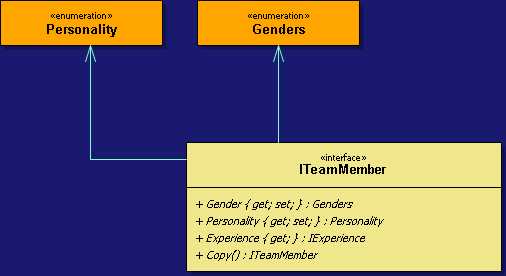
* Właściwość **ItemName** - zwraca lub zapisuje nazwę danego obiektu projektu.
* Właściwość **Color** - zwraca klasę, która przechowuje kolor danego obiektu projektu.
* Zdarzenie **ItemChanged** - wywoływane kiedy jakiekolwiek dane w obiekcie projektu zostanie zmienione. Służy głównie do poinformowania, że zaszły zmiany więc użytkownik zostanie ostrzeżony przy zamykaniu projektu o niezapisanych zmianach.
* Metoda **GetProjectItems** - jeżeli obiekt projektu jest kontenerem dla innych obiektów (tak jak np. klasa **Team** jest kontenerem dla obiektów klasy **TeamMember**) to metoda ta zwraca kolekcję przechowywanych przez ten kontener obiektów.
* Metoda **AddProjectItem(IProjectItem projectItem)** - jeżeli obiekt projektu jest kontenerem dla innych obiektów to metoda ta umożliwia dodanie kolejnego obiektu do tego kontenera.
* Metoda **RemoveProjectItem(IProjectItem projectItem)** - jeżeli obiekt projektu jest kontenerem dla innych obiektów to metoda ta umożliwa usunięcie danego obiektu z tego kontenera.
* Metoda **ClearProjectItems** - jeżeli obiekt projektu jest kontenerem dla innych obiektów to metoda ta umożliwia usunięcie wszystkich obiektów z tego kontenera.
* Metoda **CanContainProjectItem(IProjectItem projectItem)** - zwraca prawdę jeżeli dany obiekt może być przechowywany przez obiekt projektu; w innym przypadku zwraca fałsz.
* Metoda **ApplyColors** - jeżeli obiekt projektu jest kontenerem dla innych obiektów to metoda ta przepisuje swój kolor do tych obiektów.

Istnieje podstawowa implementacja tego interfejsu w abstrakcyjnej klasie **ProjectItem**.



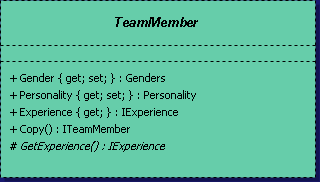
#### Interfejs ITeamMember

Interfejs **ITeamMember** definiuje podstawowe składowe członka drużyny w modelu symulacji. Dziedziczy po interfejsie **IProjectItem** i **ISimulationModel**.



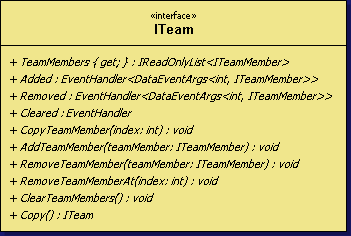
* Właściwość **Gender** - zwraca lub ustawia obiekt typu **Enum** zawierający płcie.
* Właściwość **Personality** - zwraca lub ustawia obiekt typu **Enum** zawierający cechy charakteru.
* Właściwość **Experience** - zwraca obiekt implementujący interfejs **IExperience.**
* Metoda **Copy** - zwraca kopię danego obiektu implementującego interfejs **ITeamMember**.

Istnieje podstawowa implementacja tego interfejsu w klasie abstrakcyjnej **TeamMember**. Dziedziczy po klasie **ModelItem**.



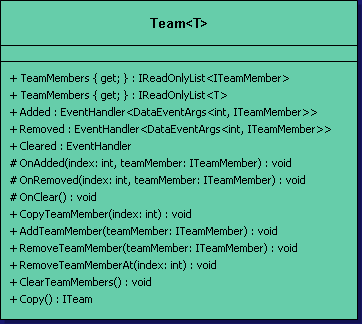
#### Intefejs ITeam

**ITeam** interfejs definiuje podstawowe elementy każdego zespołu. Dziedziczy po interfejsie **IProjectItem** i **ISimulationModel**.



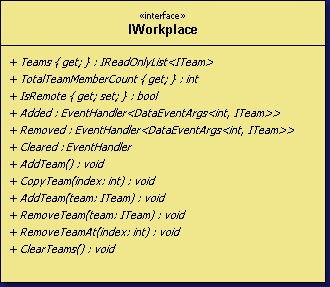
* Właściwość **TeamMembers** - zwraca listę tylko do odczytu (typ **IReadOnlyList<ITeamMember>**) zawierającą członków zespołu.
* Zdarzenie **Added** - wywoływane kiedy zostanie dodany członek zespołu.
* Zdarzenie **Removed** - wywoływane kiedy zostanie usunięty członek zespołu.
* Zdarzenie **Cleared** - wywoływane kiedy zostaną usunięci wszyscy członkowei zespołu.
* Metoda **CopyTeamMember(int index)** - kopiuje przechowywanego członka zespołu pod podaym indeksem i go dodaje do zespołu.
* Metoda **AddTeamMember(ITeamMember teamMember)** - dodaje danego członka zespołu.
* Metoda **RemoveTeamMember(ITeamMember teamMember)** - usuwa danego członka zespołu.
* Metoda **ClearTeamMembers()** - usuwa wszystkich członków zespołu.
* Metoda **Copy()** - zwraca kopię danego obiektu implementującego interfejs **ITeam**.

Istnieje podstawowa implementacja tego interfejsu w generycznej klasie **Team<T>** gdzie **T** jest typu **ITeamMember** i zawiera podstawowy bezparametrowy konstruktor. Powodem dla którego wymagany jest podstawowy bezparametrowy konstruktor jest to aby klasa **Team<T>** była wstanie w prosty sposób utworzyć członka zespołu. Dziedziczy po klasie **ModelItem**.



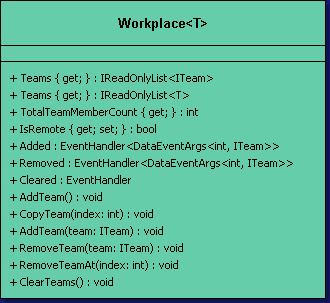
#### Intefejs IWorkplace

Interfejs **IWorkplace** definiuje obiekty reprezentujące miejsca pracy w symulacji. Dziedziczy po interfejsie **IProjectItem** i **ISimulationModel**.



* Właściwość **Teams** - zwraca listę tylko do odczytu (typ **IReadOnlyList<ITeam>**) przechowującą zespoły.
* Właściwość **TotalTeamMembersCount** - zwraca liczbę osób ze wszystkich przechowywanych zespołów.
* Właściwość **IsRemote** - zwraca prawdę jeśli miejsce pracy działa w trybie zdalnym, w innym przypadku zwraca fałsz.
* Zdarzenie **Added** - wywoływane kiedy zostanie dodany zespół.
* Zdarzenie **Removed** - wywoływane kiedy zostanie usunięty zespół.
* Zdarzenie **Cleared** - wywoływane kiedy zostaną usunięte wszystkie zespoły.
* Metoda **AddTeam()** - tworzy i dodaje nowy zespół.
* Metoda **CopyTeam(int index)** - kopiuje przechowywany zespół pod podaym indeksem i go dodaje do miejsca pracy.
* Metoda **AddTeam(ITeam team)** - dodaje dany zespół.
* Metoda **RemoveTeam(ITeam team)** - usuwa dany zespół.
* Metoda **RemoveTeamAt(int index)** - usuwa zespół pod podanym indeksem.
* Metoda **ClearTeams()** - usuwa wszystkie zespoły.

Istnieje podstawowa implementacja tego interfejsu w generycznej klasie **Workplace<T>** gdzie **T** jest typu **ITeamMember** izawiera podstawowy bezparametrowy konstruktor. Powodem dla którego wymagany jest podstawowy bezparametrowy konstruktor jest to aby klasa **Workplace<T>** była wstanie w prosty sposób utworzyć zespół. Dziedziczy po klasie **ModelItem**.



#### Interfejs IExperience

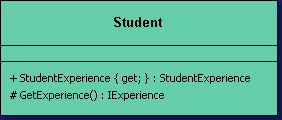
Każdy członek zespołu ma jakieś danego typu doświadczenie, które jest reprezentowane przez interfejs **IExperience**.



* Właściwość **Age** - zwraca lub ustawia wiek członka zespołu.
* Zdarzenie **Changed** - wywołuje się kiedy dane w obiekcie implementującym **IExperience** zostały zmodyfikowane.

#### Klasa Student

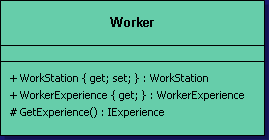
Klasa **Student** reprezentuje studenta na uczelni w symulacji. Dziedziczy po klasie **TeamMember.**



* Właściwość **StudentExperience** - zwraca obiekt **StudentExperience** przechowujący dane o doświadczeniu studenta.
* Metoda **GetExperience()** - zwraca właściwość **StudentExperience** dla właściwości **Experience** z interfejsu **ITeamMember**.

#### Klasa Worker

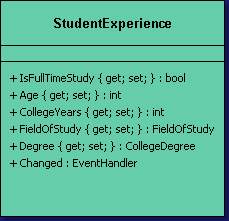
Klasa **Worker** reprezentuje nracownika w firmie w symulacji. Dziedziczy po klasie **TeamMember.**



* Właściwość **WorkerExperience** - zwraca obiekt **WorkerExperience** przechowujący dane o doświadczeniu pracownika.
* Właściwość **WorkStation** - zwraca lub modyfikuje obiekt typu **Enum** przechowujący stanowisko pracy pracownika.
* Metoda **GetExperience()** - zwraca właściwość **WorkerExperience** dla właściwości **Experience** z interfejsu **ITeamMember**.

#### Klasa StudentExperience

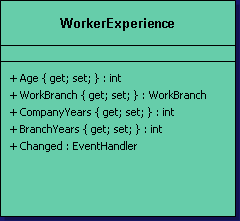
Klasa **StudentExperience** zawiera dane o doświadczeniu studenta, jest wykorzystywana przez klasę **Student**. Implementuje interfejs **IExperience**.



* Właściwość **IsFullTimeStudy** - zwraca lub modyfikuje flagę informującą czy student studiuje w trybie stacjonarnym czy niestacjonarnym.
* Właściwość **Age** - patrz **IExperience.Age**.
* Właściwość **CollegeYears** - zwraca lub modyfikuje ile lat student studiuje na uczelni.
* Właściwość **FieldOfStudy** - zwraca lub modyfikuje obiekt typu **Enum** przechowujący kierunek studiów.
* Właściwość **Degree** - zwraca lub modyfikuje obiekt typu **Enum** przechowujący stopień studiów.
* Zdarzenie **Changed** - patrz **IExperience.Changed**.

#### Klasa WorkerExperience

Klasa **WorkerExperience** zawiera dane o doświadczeniu pracownika, jest wykorzystywana przez klasę **Worker**. Implementuje interfejs **IExperience**.



* Właściwość **Age** - patrz **IExperience.Age**.
* Właściwość **WorkBranch** - zwraca lub modyfikuje obiekt typu **Enum** przechowujący branżę pracy.
* Właściwość **CompanyYears** - zwraca lub modyfikuje ile lat pracownik pracuje w danej firmie.
* Właściwość **BranchYears** - zwraca lub modyfikuje ile lat pracownik pracuje w danej branży.
* Zdarzenie **Changed** - patrz **IExperience.Changed**.

#### Klasa Univercity

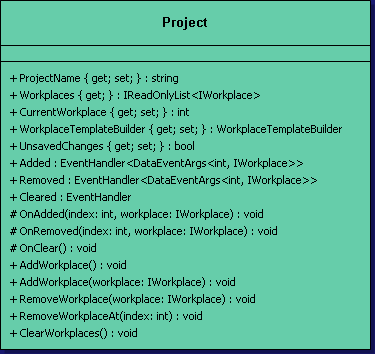
Klasa **Univercity** jedynie dziedziczy po klasie **Workplace<T>** gdzie generyczny parametr **T** jest typu **Student**.

#### Klasa Company

Klasa **Company** jedynie dziedziczy po klasie **Workplace<T>** gdzie generyczny parametr **T** jest typu **Worker**.

#### Klasa Project

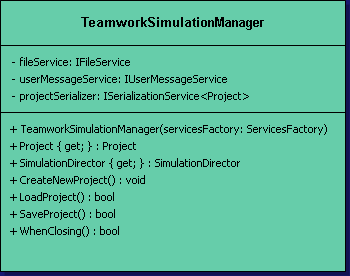
Klasa **Project** jest głównym elementem w drzewie projektu przechowującym wszystkie informacje potrzebne dla symulacji. Dziedziczy po klasie **ModelItem**.



* Właściwość **ProjectName** - zwraca lub modyfikuje nazwę projektu.
* Właściwość **Workplaces** - zwraca listę tylko do odczytu (typu **IReadOnlyList<IWorkplace>**) zawierającą wszystkie miejsca pracy projektu.
* Właściwość **CurrentWorkplace** - zwraca lub modyfikuje indeks aktualnego miejsca pracy, przy którym użytkownik pracuje.
* Właściwość **WorkplaceTemplateBuilder** - zwraca lub modyfikuje klasę **WorkplaceTemplateBuilder**, która tworzy klasy typu **IWorkpalce**. Pozwala to na elastyczną zmianę procesu tworzenia miejsc pracy różnego typu.
* Właściwość **UnsavedChanges** - zwraca lub modyfikuje flagę wskazującą czy w projekcie są niezapisane zmiany.
* Zdarzenie **Added** - wywołuje się kiedy zostanie dodane miejsce pracy.
* Zdarzenie **Removed** - wywołuje sie kiedy zostanie usunięte miejsce pracy.
* Zdarzenie **Cleared** - wywołuje się kiedy zostaną usunięte wszystkie miejsca pracy.
* Metoda **OnAdded** - “protected” metoda wywołująca zdarzenie **Added**.
* Metoda **OnRemoved** - “protected” metoda wywołująca zdarzenie **Removed**.
* Metoda **OnCleared** - “protected” metoda wywołująca zdarzenie **Cleared**.
* Metoda **AddWorkplace** - dodaje nowe miejsce pracy zgodnie z obiektem z właściwości **WorkplaceTemplateBuilder**.
* Metoda **AddWorkplace(IWorkplace workplace)** - dodaje dane miejsce pracy.
* Metoda **RemoveWorkplace(IWorkplace workplace)** - usuwa dane miejsce pracy.
* Metoda **RemoveWorkplaceAt(int index)** - usuwa miejsce pracy pod podanym indeksem.
* Metoda **ClearWorkplaces()** - usuwa wszystkie miejsca pracy.

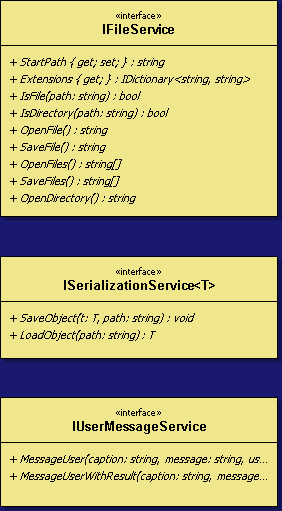
#### Klasa TeamworkSimulationManager

**TeamworkSimulationManager** jest główną klasą modelu interfejsu. To ona inicjalizuje wszystkie potrzebne obiekty.



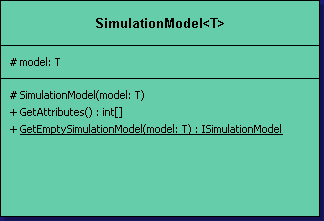
* Pole **fileService** - zawiera obiekt typu **IFileService** potrzebny do działania na plikach
* Pole **userMessageService** - zawiera obiekt typu **IUserMessageService** potrzebny do informowania użytkownika o pewnych zdarzeniach.
* Pole **projectSerializer** - zawiera obiekt typu **ISerializationService<Project>** pozwalający na zapisywanie i wczytywanie obiektów klasy **Project** i jej składowych.
* Konstruktor – przyjmuje obiekt klasy **ServicesFactory**, która to przechowuje odpowiednie typy serwisów dla danej konfiguracji.
* Właściwość **Project** - zwraca obiekt typu **Project**.
* Właściwość **SimulationDirector** - zwraca obiekt typu **SimulationDirector** do integracji z symulacją.
* Metoda **CreateNewProject()** - tworzy nowy projekt modyfikując przy tym właściwość **Project**.
* Metoda **LoadProject()** - ładuje projekt z pliku, który jest wybierany przez użytkownika poprzez **fileService**.
* Metoda **SaveProject()** - zapisuje aktualny projekt do pliku, który jest wybierany przez użytkownika poprzez **fileService**.
* Metoda **WhenClosing()** - wywoływana kiedy aplikacja jest zamykana. Jeśli istnieją niezapisane zmiany metoda wykorzysta **userMessageService** do poinformowania o niezapisanych zmianach i możliwości zapisu tych zmian.

#### Interfejsy serwisów



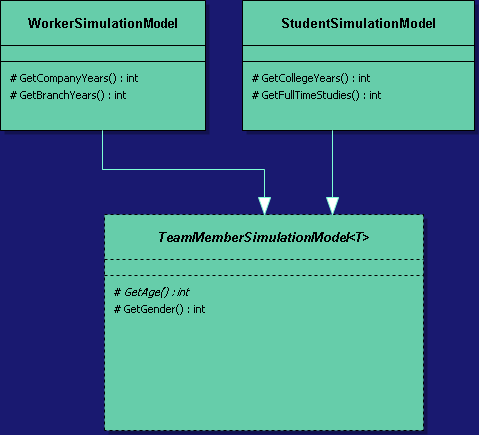
#### Klasa SimulationModel<T>

Generyczna klasa **SimulationModel<T>**,gdzie **T** to dowolny obiekt, implementuje interfejs **ISimulationModel**.

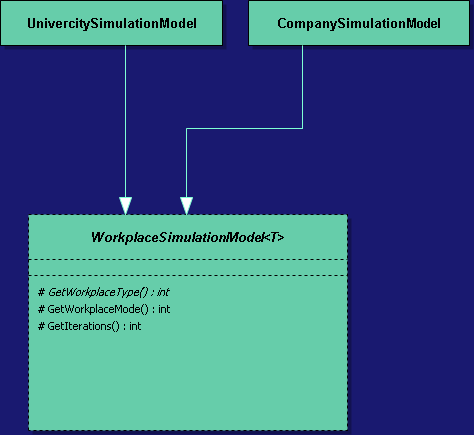


* Pole **model** - przechowuje obiekt, względem którego zostanie zwrócona odpowiednia tablica dla symulacji.
* Konstruktor – przyjmuje obiekt typu **T**,który zostanie zapisany w polu **model**.
* Metoda **GetEmptySimulationModel(T model)** - statyczna metoda, która zwraca pusty obiekt typu **ISimulationModel** dla tych klas, które nie zawierają informacji dla symulacji.

#### Klasy TeamMemberSimulationModel



#### Klasy WorkplaceSimulationModel



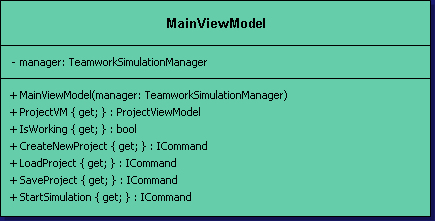
### Widok Modelu (View Model)

#### Klasa NotifyPropertyChanges

Jest to klasa abstrakcyjna, po której dziedziczy każda klasa widoku modelu. Implementuje ona interfejs **INotifyPropertyChanges** do informowania jakie składniki modelu się zmieniły.

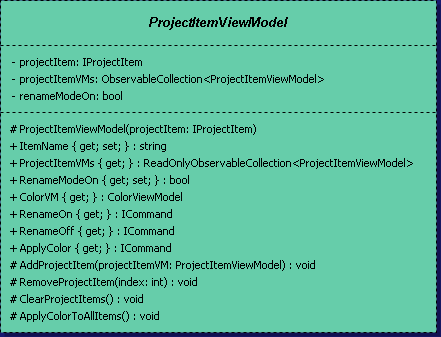
#### Klasa MainViewModel

Klasa ta jest widokiem modelu klasy **TeamworkSimulationManager**.



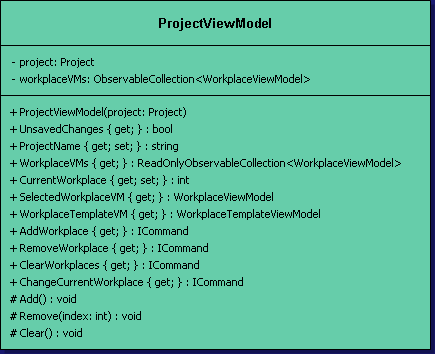
#### Klasa ProjectItemViewModel

Klasa ta jest widokiem modelu klasy **ProjectItem**. Dziedziczy po **NotifyPropertyChanges**.



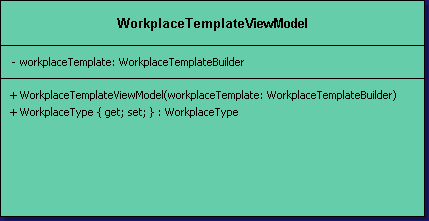
#### Klasa ProjectViewModel

Klasa **ProjectViewModel** jest widokiem modelu klasy **Project**. Dziedziczy po **ProjectItemViewModel**.



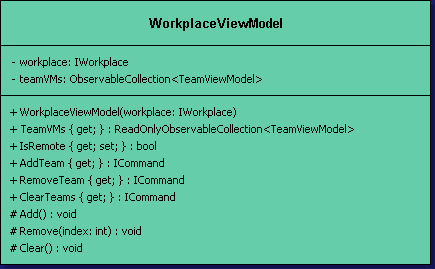
#### Klasa WorkplaceTemplateViewModel

Klasa ta jest widokiem modelu klasy **WorkplaceTemplateBuilder**. Dziedziczy po **ProjectItemViewModel**.



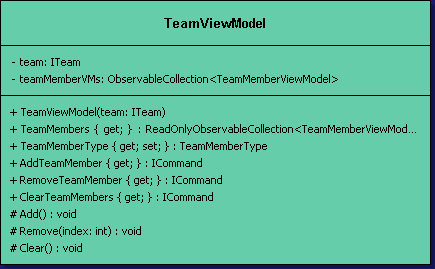
#### Klasa WorkplaceViewModel

Klasa ta jest widokiem modelu obiektów implementujących interfejs **IWorkplace**. Dziedziczy po **ProjectItemViewModel**.



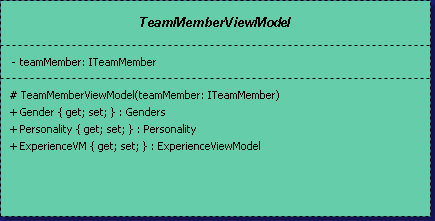
#### Klasa TeamViewModel

Klasa ta jest widokiem modelu obiektów implementujących interfejs **ITeam**. Dziedziczy po **ProjectItemViewModel**.



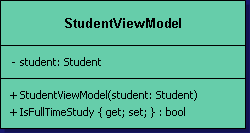
#### Klasa TeamMemberViewModel

Klasa abstrakcyjna, która jest widokiem modelu obiektów implementujących interfejs **ITeamMember**. Dziedziczy po **ProjectItemViewModel**.



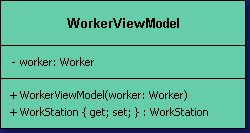
#### Klasa StudentViewModel

Klasa, która jest widokiem modelu obiektów klasy **Student**.



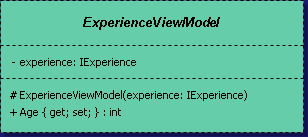
#### Klasa WorkerViewModel

Klasa, która jest widokiem modelu obiektów klasy **Worker**.



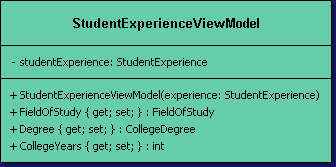
#### Klasa ExperienceViewModel

Abstrakcyjna klasa, która jest widokiem modelu obiektów implementujących interfejs **IExperience**. Dziedziczy po **ProjectItemViewModel**.



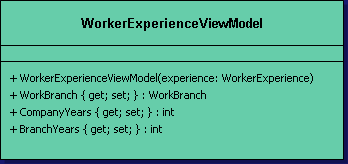
#### Klasa StudentExperienceViewModel

Klasa, która jest widokiem modelu obiektów klasy **StudentExperience**.



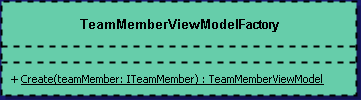
#### Klasa WorkerExperienceViewModel

Klasa, która jest widokiem modelu obiektów klasy **WorkerExperience**.



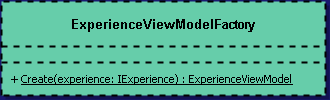
#### Klasa TeamMemberViewModelFactory

Klasa statyczna, która tworzy widoki modelu zgodnie z typem modelu **ITeamMember**.



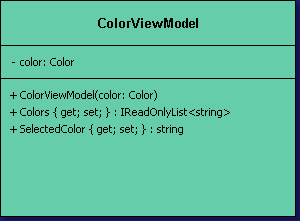
#### Klasa ExperienceViewModelFactory

Klasa statyczna, która tworzy widoki modelu zgodnie z typem modelu **IExperience**.



#### Klasa ColorViewModel.

Klasa, która jest widokiem modelu obiektów klasy **Color**. Dziedziczy po **ProjectItemViewModel**.



**Elementy do poprawy w przyszłym semestrze:**

-Optymalizacja funkcji obliczania prawdopodobieństwa w modelu

-Dostrajanie modelu na podstawie danych z ankiet