



UNIWERSYTET KAZIMIERZA WIELKIEGO W BYDGOSZCZY

Dokumentacja Techniczna Systemu Eksperskiego
Wspomagającego Wybór postaci do kampanii D&D.

Inżynieria Wiedzy i Systemy Ekspertowe

Sebastian Mrowiński

Informatyka III rok

Index:97727

Spis treści

1. Ogólne założenia systemu	3
2. Schemat działania.....	4
Wybór Rasy:.....	4
Wybór Klasy:.....	4
Stworzenie Postaci:	4
Podsumowanie Gotowej Postaci:.....	4
Przykład wynikowego podsumowania:	4
3. Właściwości aplikacji: źródła, fasety.....	5
Baza Metafor:	6
Baza Wyjaśnień CO TO JEST?:.....	6
Ekspercka Baza Wiedzy:	6
Fasety:	6
Napotkane Trudności:	6
4. Terminologia i stałe wartości	7
SYSTEM:.....	7
UŻYTKOWNIK:.....	7
PYTANIA:.....	7
Wartości Preferowane przez System:	7
5. Materiały multimedialne i objaśnienia.....	8
Grafika:	8
Nowatorski Format Grafiki:	8
Źródła Wizualne:.....	8
Objaśnienia i metafory:	8
Baza Wyjaśnień CO TO JEST?:.....	8
Baza Metafor:	9
6. Schemat działania — drzewa decyzyjne.....	10
Baza Reguł źródeł wiedzy:	10
Konkluzja postaci „Dragonborn Rouge” w programie CAKE:	12
Drzewo Systemu tworzenia Postaci w D&D:	12
7. DeTreex	14
8. Implementacja systemu ekspertowego w języku Python (automatyczny).....	16
Biblioteki:.....	17
Wczytanie Danych:	17

Label Encoding:.....	17
Budowa Drzewa Decyzyjnego:	17
Interfejs Graficzny:	17
Przewidywanie Postaci:	17
Aktualizacja Opisów:	17
Wynik działania programu:	18
9. Implementacja systemu ekspertowego w języku Python (ręczny)	19
Biblioteki i Interfejs Graficzny:	20
Wczytywanie Opisów:	20
Reguły Systemu Ekspertckiego:	20
Prezentacja Wyników:.....	20
Zakończenie Programu:	20
Działanie Reguł:	21
Interaktywność:.....	21
10. Porównanie implementacji SPHINX i Python	21

1. Ogólne założenia systemu



Prosty i przyjazny dla użytkownika interfejs systemu ekspertckiego, który opiera się na serii kilku pytań, został zaprojektowany w celu ułatwienia procesu

tworzenia postaci dla graczy D&D. Główny nacisk kładziony jest na wybór rasy i klasy postaci - dwóch kluczowych aspektów postaci w uniwersum gry.

System wykorzystuje modele wnioskowania i reguły oparte na wyborach gracza, stylu gry i ogólnych oczekiwaniach wobec postaci, korzystając z bazy wiedzy odnoszącej się do uniwersum D&D. Celem tej metody jest dopasowanie gustów i wymagań użytkownika do odpowiednich ras i klas postaci.

Podobnie jak inne narzędzia tego typu, system ekspercki ma na celu ułatwienie graczom podejmowania decyzji, aby mogli szybko i skutecznie stworzyć postać, która będzie reprezentować ich wyobrażenie o tym, jak wygląda bohater w uniwersum D&D. Ponad to dodatkowe opisy i grafiki dodane do systemu eksperckiego pozwala na większe zrozumienie jakie gracz dokonuje podczas jego używania.

2. Schemat działania

Wybór Rasy:

- Gracz wybiera rasę spośród dostępnych opcji (np. Human, Elf, Dwarf, Dragonborn, Tiefling).
- Na podstawie wyboru przedstawiany jest opis rasy.

Wybór Klasy:

- Gracz wybiera klasę postaci spośród dostępnych opcji (np. Fighter, Wizard, Rogue, Barbarian).
- Na podstawie wyboru przedstawiany jest opis klasy.

Stworzenie Postaci:

- Na podstawie wybranych rasy i klasy wybierana jest najbardziej pasująca postać dla gracza.

Podsumowanie Gotowej Postaci:

- System prezentuje gotową postać, uwzględniając imię, rasę, klasę oraz krótki opis.

Przykład wynikowego podsumowania:

Stworzona Postać!

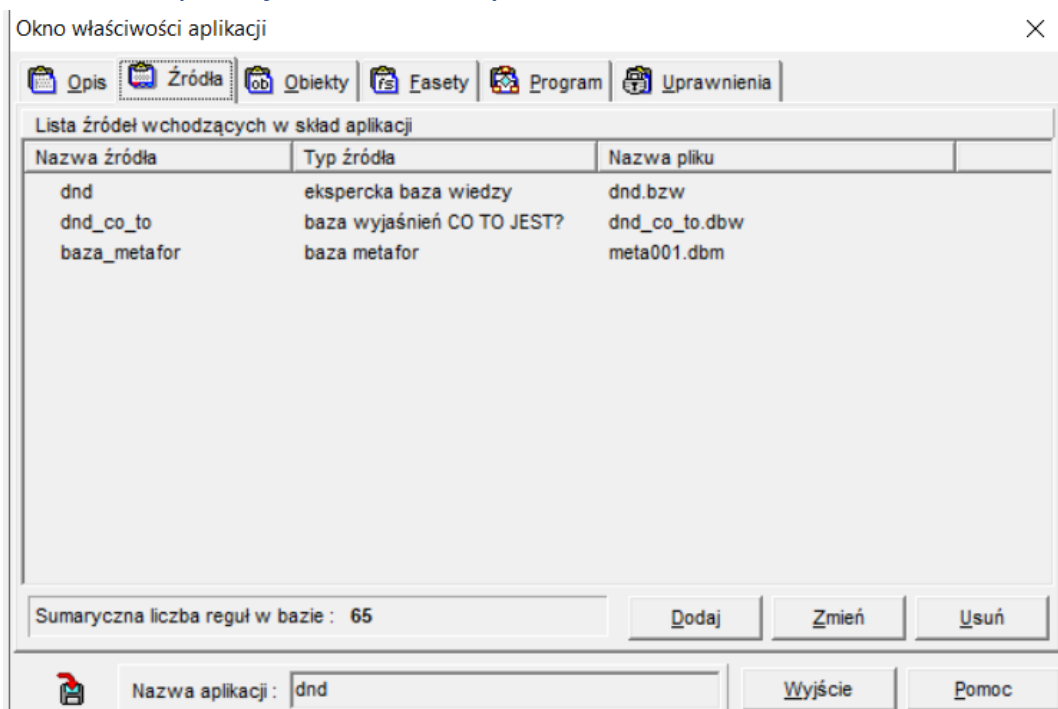
Imię: Roland Barns

Rasa: Dragonborn

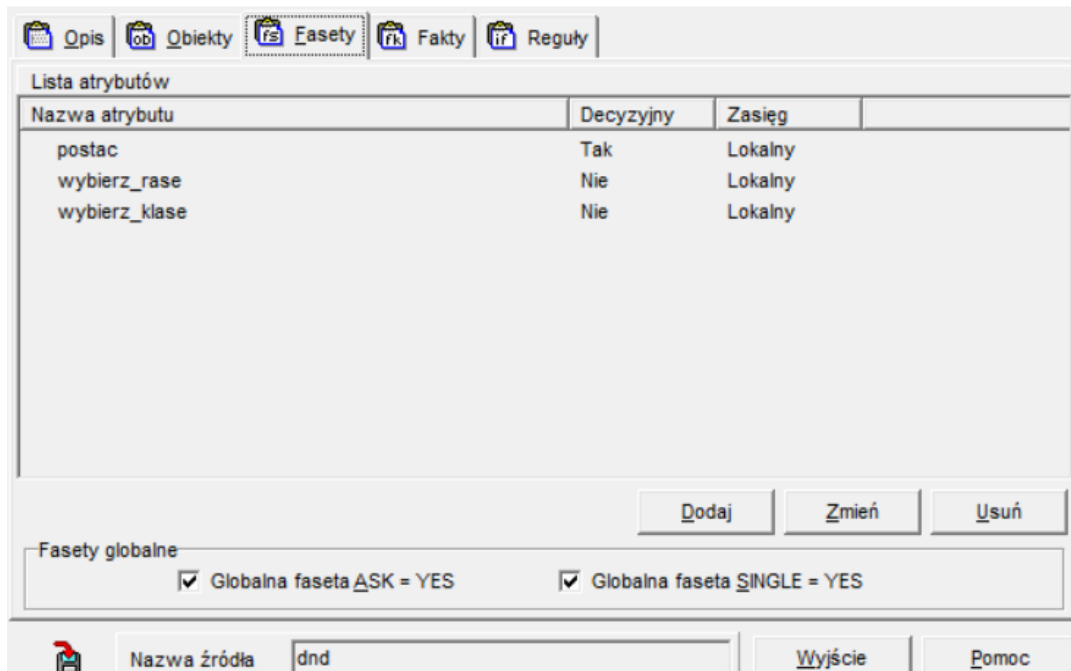
Klasa: Barbarian

Historia: Smoczy wojownik który opuścił swoje rodzinne miasto Nemfis w poszukiwaniu przygód.

3. Właściwości aplikacji: źródła, fasety



Okno właściwości eksperckiego źródła wiedzy



Nazwa atrybutu	Decyzyjny	Zasięg
postac	Tak	Lokalny
wybierz_rase	Nie	Lokalny
wybierz_klase	Nie	Lokalny

Fasety globalne
☒ Globalna faseta ASK = YES
 ☒ Globalna faseta SINGLE = YES

Nazwa źródła: dnd

Baza Metafor:

- Stanowi fundament systemu, definiując zasady.

Baza Wyjaśnień CO TO JEST?:

- Pomaga użytkownikowi zrozumieć związki między wyborami, ułatwiając świadome podejmowanie decyzji podczas tworzenia postaci.

Ekspercka Baza Wiedzy:

- Obejmuje podstawową wiedzę ekspertów na temat tworzenia postaci w D&D.

Fasety:

- Opis: Zbiór kluczowych atrybutów w bazie wiedzy, umożliwiający precyzyjne dopasowanie postaci do preferencji gracza.

Napotkane Trudności:

- Problem techniczny związany z bazą metafor nie został rozwiązany. Aplikacja utrudnia dostęp do danych mimo istnienia pliku oraz nie pozwala na jego usunięcie w celach poprawy.

4. Terminologia i stałe wartości

SYSTEM:

- System ekspercki wspomagający tworzenia postaci w świecie D&D, uwzględniający preferencje gracza.

UŻYTKOWNIK:

- Osoba korzystająca z aplikacji do tworzenia postaci, dokonująca wyborów dotyczących rasy, klasy zgodnie z własnymi preferencjami.

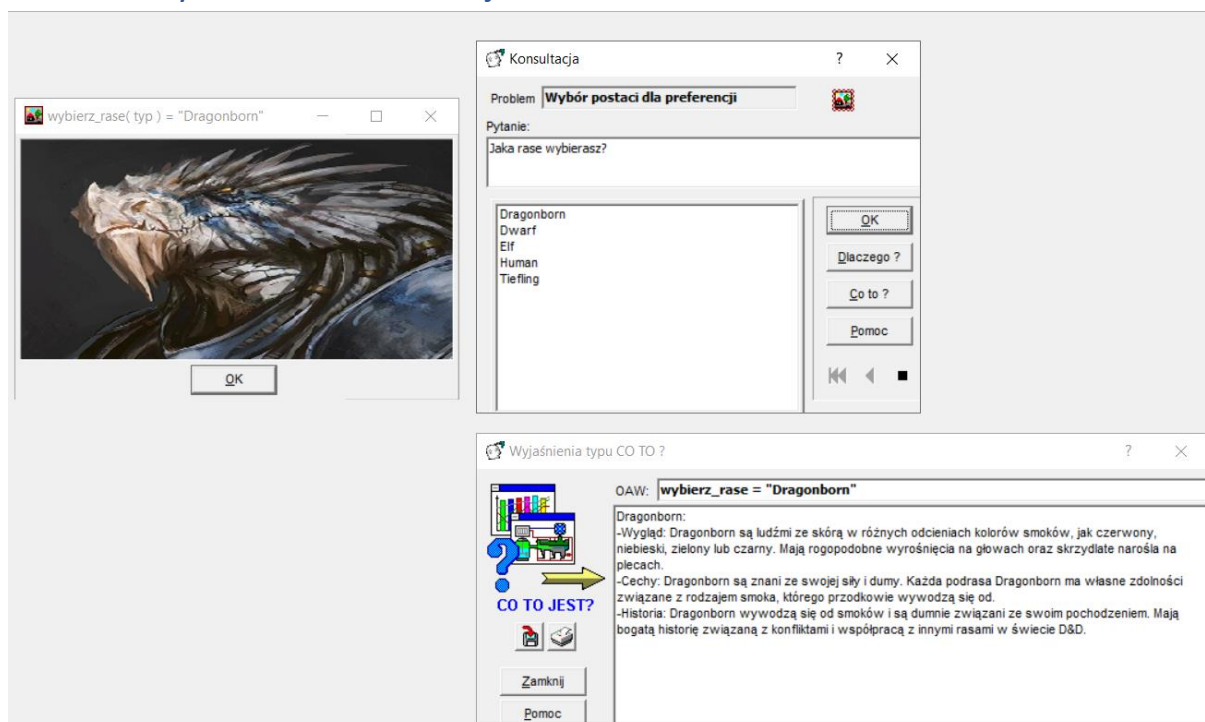
PYTANIA:

- Seria kilku pytań, które użytkownik musi odpowiedzieć w celu dostosowania postaci do swoich preferencji. Pytania obejmują wybór rasy oraz klasy.

Wartości Preferowane przez System:

- *Jaka rase wybierasz:* Dragonborn, Dwarf, Elf, Human, Tiefling.
- *Jaki typ postaci Cie interesuje:* Furia i Wściekłość, Mistrz Miecza, Sztuki Walki i Duchowa Energia, Straznik Swiatla, Infiltracja i Obrazenia krytyczne, Wsparcie i Leczenie, Inżynieria magiczna, Wiedza i Arkana Magi, Narodzony z Magia, Muzyka i Magia, Zrecznosc i Bron dystansowa, Magia paktow i Cienie, Przywiązanie do natury.

5. Materiały multimedialne i objaśnienia



Grafika:

- Aplikacja została wzbogacona o elementy wizualne, takie jak zdjęcia, aby zwiększyć zrozumienie i atrakcyjność pytań dotyczących tworzenia postaci.

Nowatorski Format Grafiki:

- Pliki graficzne zostały przekonwertowane na format BMP, co pozwala na efektywne ich wykorzystanie w systemie.

Źródła Wizualne:

- Materiały graficzne pochodzą głównie z Google Images, ponieważ posiada największą różnorodność dzięki czemu elementy wizualne są spójne i różnorodne.

Objaśnienia i metafory:

W moim programie do tworzenia postaci do D&D, objaśnienia i metafory są kluczowym elementem, który pomaga użytkownikom zrozumieć, dlaczego system podejmuje konkretne decyzje dotyczące postaci.

Baza Wyjaśnień CO TO JEST?:

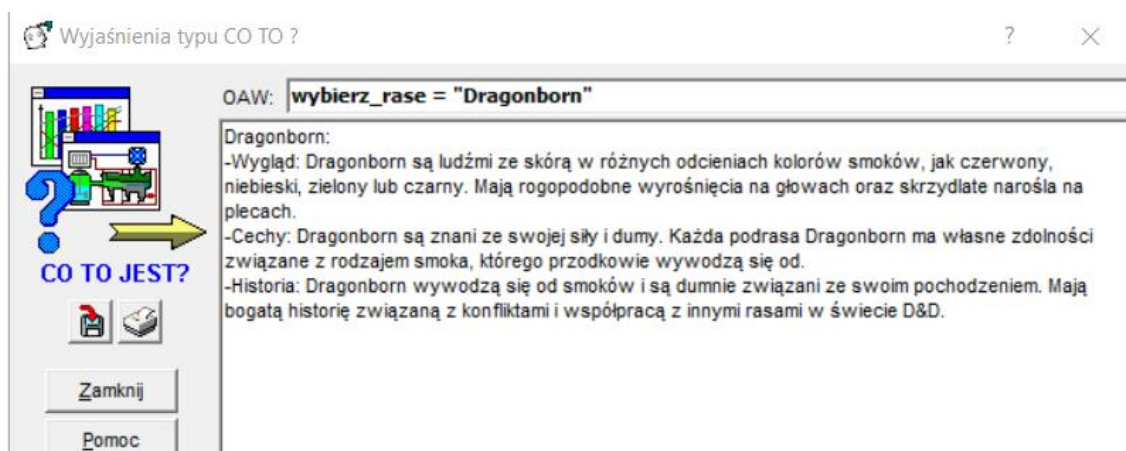
- Wyjaśnienia tłumaczą, dlaczego system wybrał daną postać na podstawie udzielonych przez użytkownika odpowiedzi.

Baza Metafor:

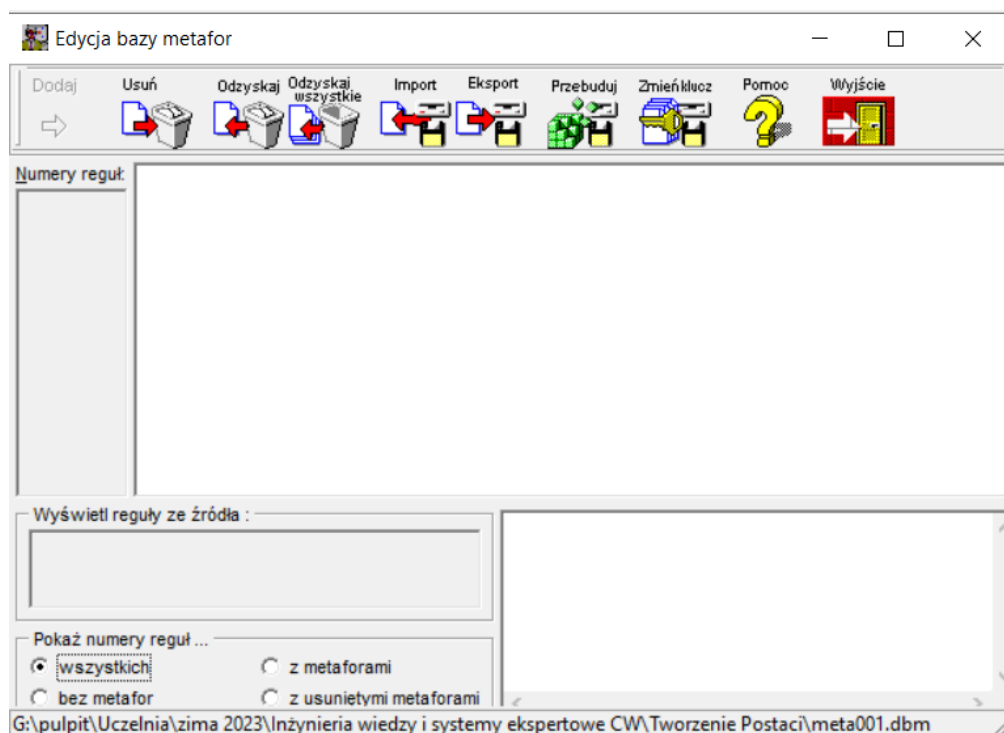
- Metafory opisują reguły systemu w bardziej zrozumiały sposób, pomagając użytkownikowi lepiej zrozumieć, jakie zasady kierują procesem tworzenia postaci.

Te dwie bazy wiedzy działają razem, aby umożliwić użytkownikom pełniejsze zrozumienie decyzji systemu. Po zakończeniu interakcji z aplikacją, użytkownicy mogą szczegółowo przeanalizować, jakie reguły i metafory wpłynęły na stworzenie konkretnej postaci w świecie D&D.

Przykład wyjaśnienia - CO TO?



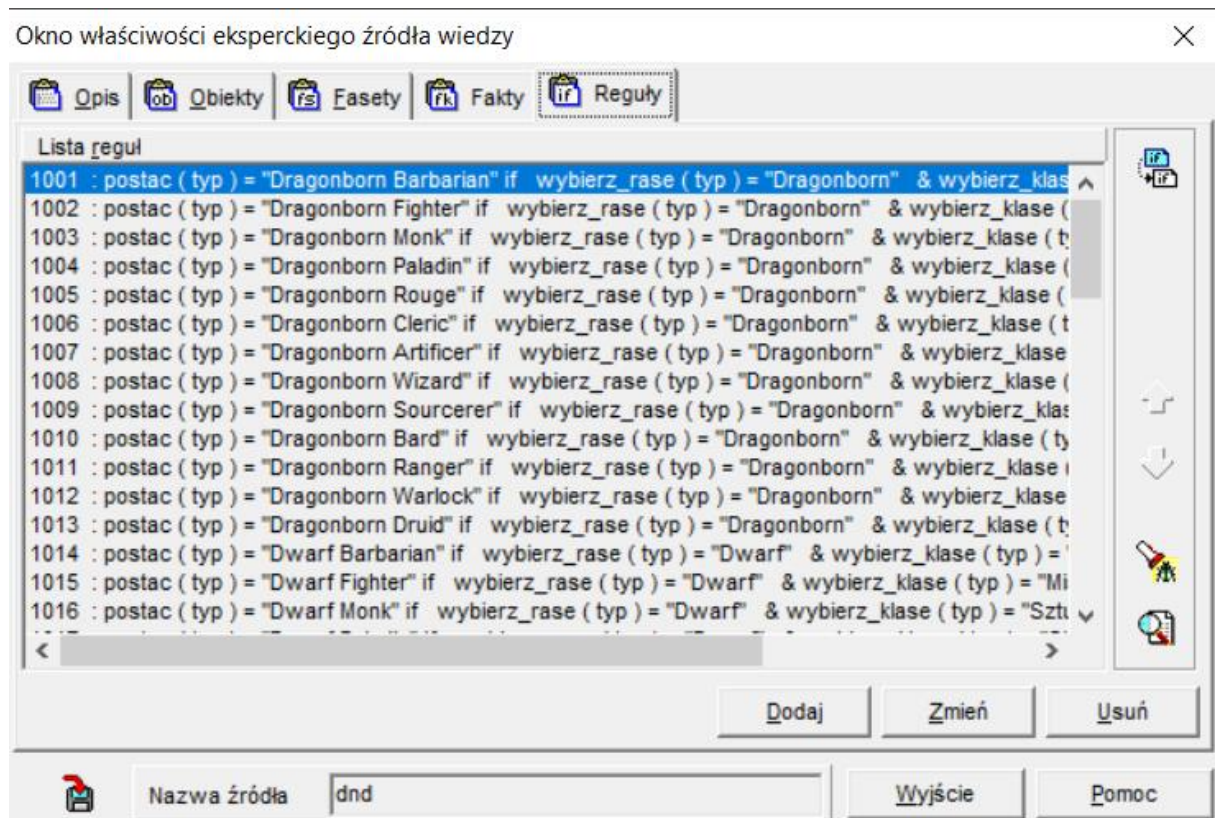
Przykład wyjaśnienia – baza metafor



Niestety baza metafor ma problemy przez które nie byłem w stanie przejść.

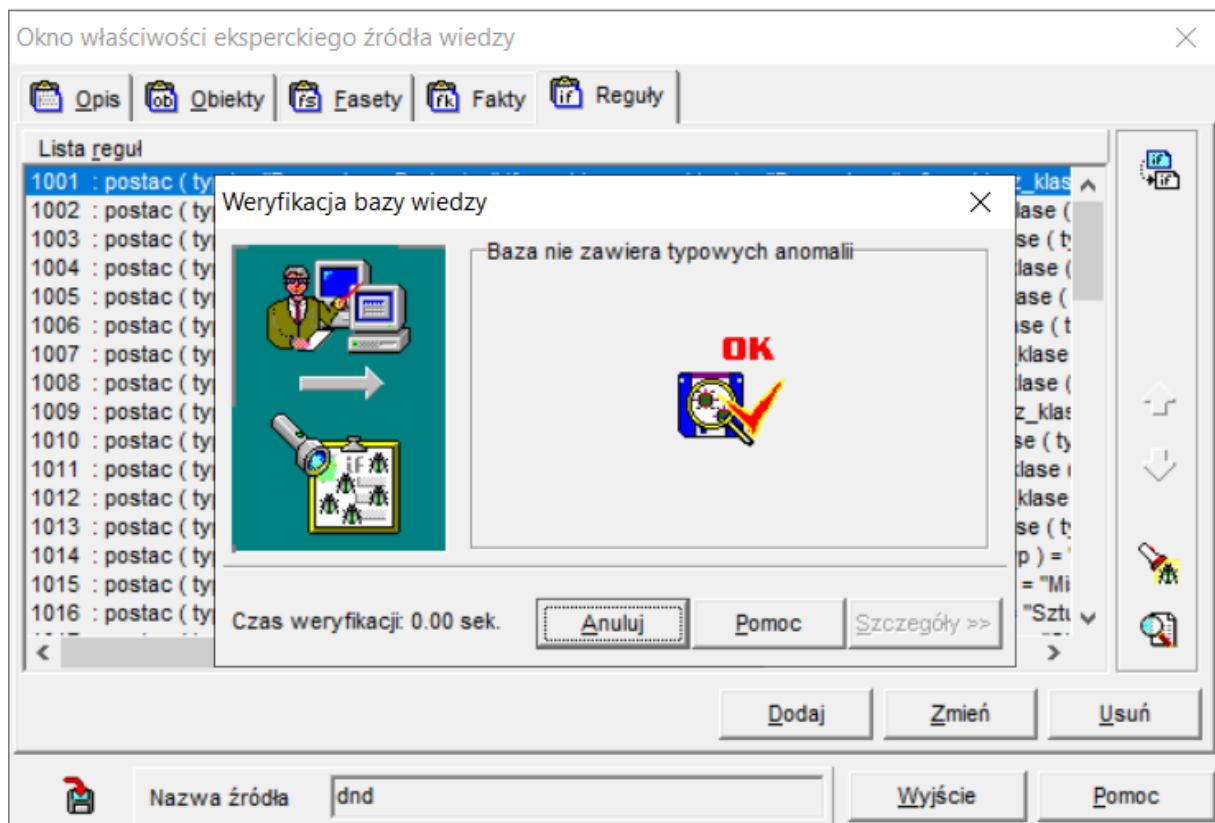
6. Schemat działania — drzewa decyzyjne

Drzewa decyzyjne to jeden ze sposobów na wizualne przedstawienie wspomaganie decyzji w systemie.

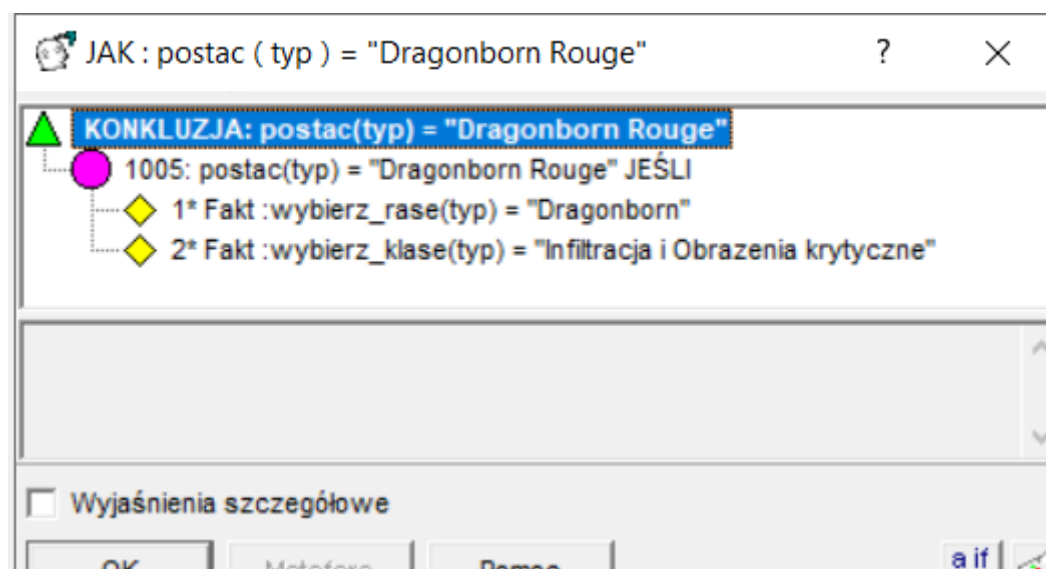


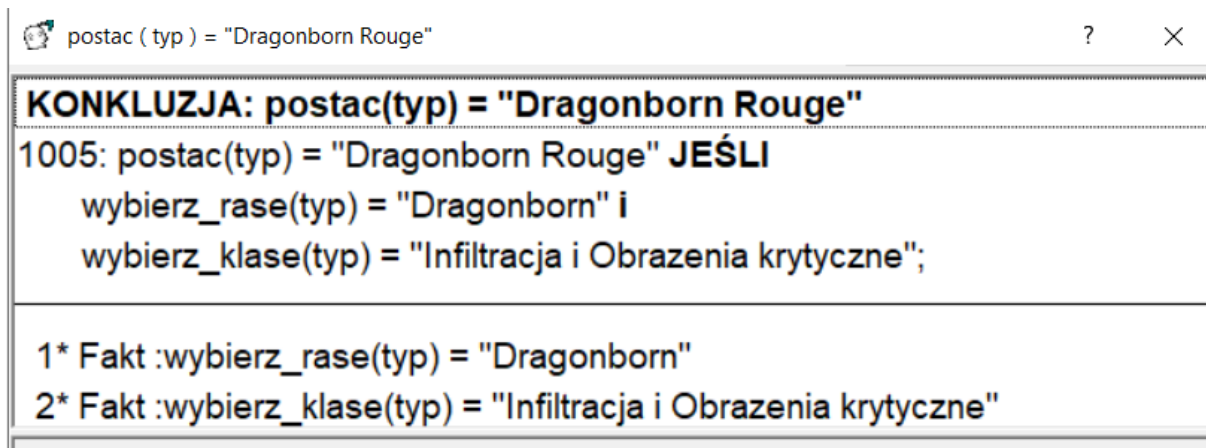
Baza Reguł źródeł wiedzy:

W systemie eksperckim do tworzenia postaci w D&D, Baza Reguł stanowi fundament zaimplementowany zgodnie z przemyślanym drzewem decyzyjnym. Reguły są opisane przy użyciu trójki OAW (Obiekt-Atrybut-Wartość), co jest podstawową strukturą reprezentacji wiedzy w tym systemie. Każdy powiązany atrybut tworzy fasetę, która precyzyjnie definiuje cechy postaci.



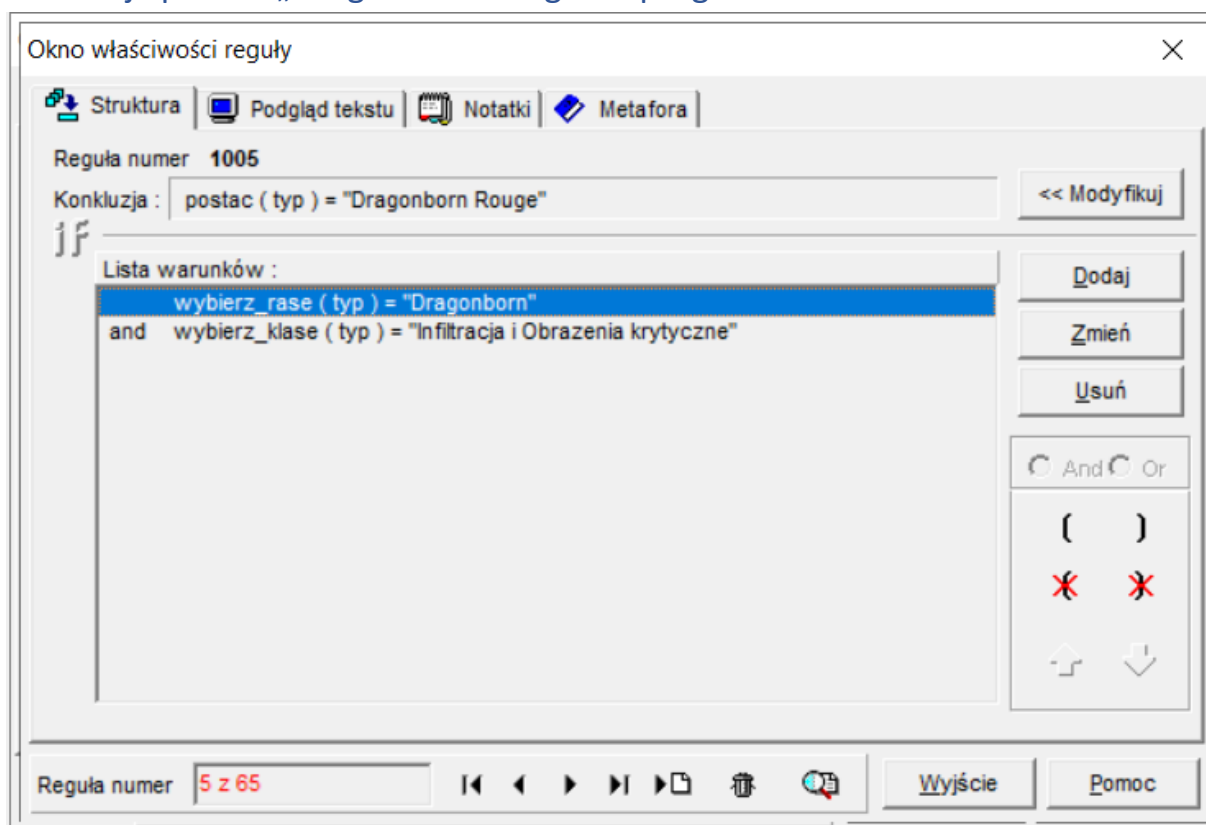
Baza wiedzy została pomyślnie zweryfikowana. Wszystko zostało wprowadzone poprawnie.





Wnioskowanie w systemie generuje konkluzje, czyli rezultaty decyzji. Te konkluzje przedstawiają schemat instrukcji warunkowych, które kierują procesem tworzenia postaci. Dzięki temu, użytkownicy mogą zobaczyć, jakie reguły i atrybuty wpływają na ostateczny wybór postaci.

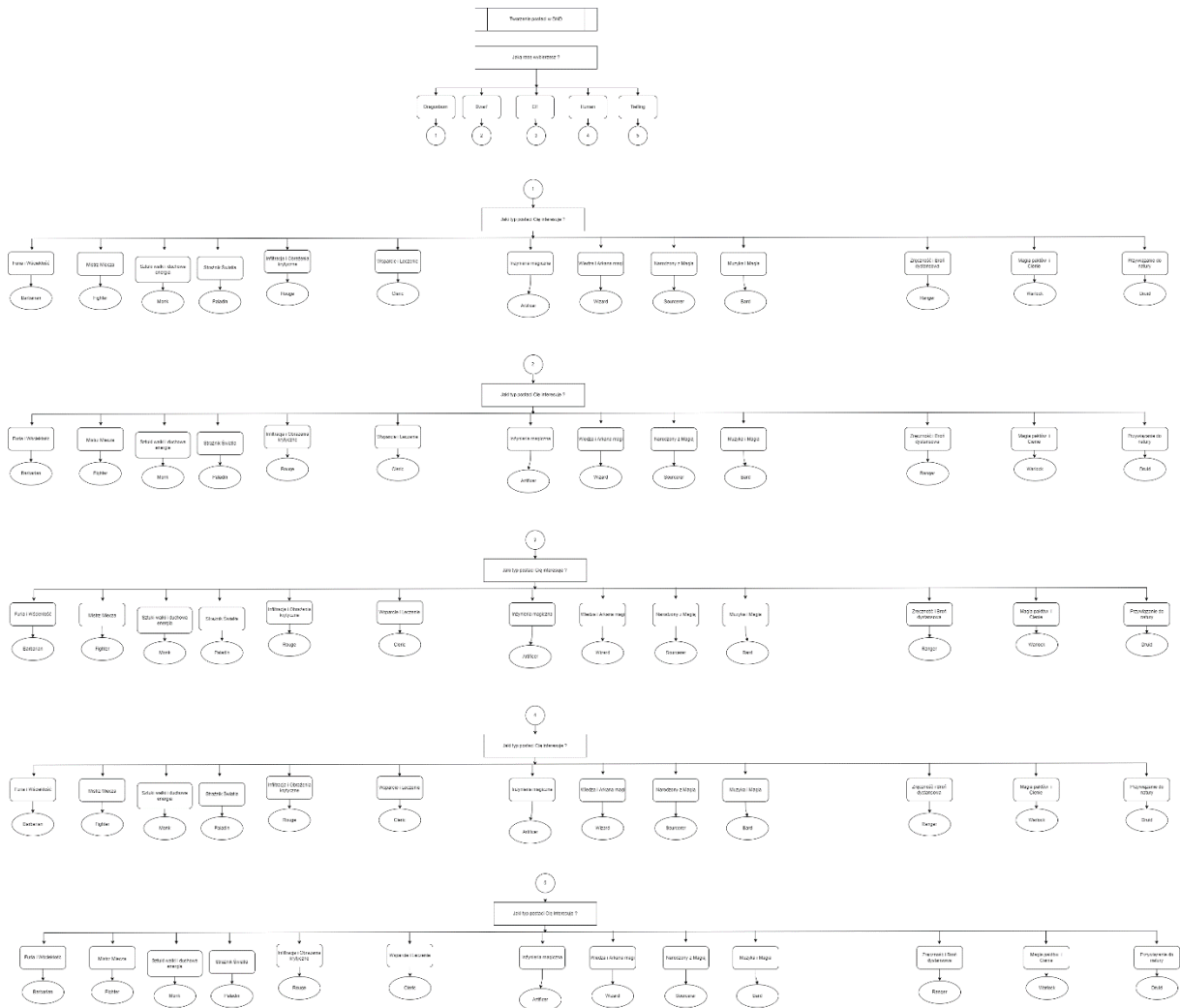
Konkluzja postaci „Dragonborn Rouge” w programie CAKE:



Drzewo Systemu tworzenia Postaci w D&D:

W systemie eksperckim do tworzenia postaci w D&D, drzewo systemu wyboru zostało rozbite na mniejsze segmenty, co pozwala na lepsze zrozumienie jego założeń. Ten schemat decyzyjny został stworzony przy użyciu narzędzia na

stronie draw.io, co pozwoliło na precyzyjne zobrazowanie kroków wyboru postaci w systemie. Każdy fragment drzewa reprezentuje konkretne pytania i decyzje, które użytkownik podejmuje, prowadząc do ostatecznego stworzenia postaci w świecie D&D.



7. DeTreeex

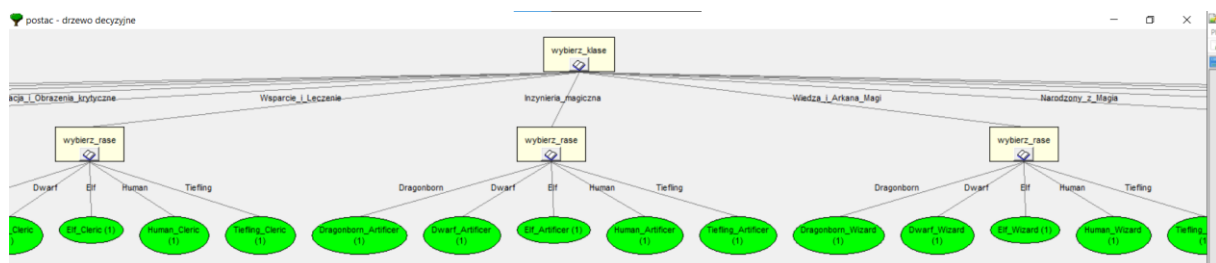
Plik uczący w Excelu

	A	B	C	D
1				
2		we	we	wy
3		#wybierz rase	#wybierz klase	#postac
4		Dragonborn	Furia i Wscieklosc	Dragonborn Barbarian
5		Dragonborn	Mistrz Miecza	Dragonborn Fighter
6		Dragonborn	Sztuki walki i duchowa energia	Dragonborn Monk
7		Dragonborn	Straznik Swiatla	Dragonborn Paladin
8		Dragonborn	Infiltracja i Obrazenia krytyczne	Dragonborn Rouge
9		Dragonborn	Wsparcie i Leczenie	Dragonborn Cleric
10		Dragonborn	Inzynieria magiczna	Dragonborn Artificer
11		Dragonborn	Wiedza i Arkana Magi	Dragonborn Wizard
12		Dragonborn	Narodzony z Magia	Dragonborn Sourcerer
13		Dragonborn	Muzyka i Magia	Dragonborn Bard
14		Dragonborn	Zrecznosc i Bron dystansowa	Dragonborn Ranger
15		Dragonborn	Magia paktow i Cienie	Dragonborn Warlock
16		Dragonborn	Przywiazanie do natury	Dragonborn Druid
17		Dwarf	Furia i Wscieklosc	Dwarf Barbarian
18		Dwarf	Mistrz Miecza	Dwarf Fighter
19		Dwarf	Sztuki walki i duchowa energia	Dwarf Monk
20		Dwarf	Straznik Swiatla	Dwarf Paladin
21		Dwarf	Infiltracja i Obrazenia krytyczne	Dwarf Rogue
22		Dwarf	Wsparcie i Leczenie	Dwarf Cleric
23		Dwarf	Inzynieria magiczna	Dwarf Artificer
24		Dwarf	Wiedza i Arkana Magi	Dwarf Wizard
25		Dwarf	Narodzony z Magia	Dwarf Sorcerer
26		Dwarf	Muzyka i Magia	Dwarf Bard
27		Dwarf	Zrecznosc i Bron dystansowa	Dwarf Ranger
28		Dwarf	Magia paktow i Cienie	Dwarf Warlock
29		Dwarf	Przywiazanie do natury	Dwarf Druid

Plik uczący lrm

```
dnd.lrm
1 we we wy
2 #wybierz_rase #wybierz_klase #postac
3 Dragonborn Furia_i_Wscieklosc Dragonborn_Barbarian
4 Dragonborn Mistrz_Miecza Dragonborn_Fighter
5 Dragonborn Sztuki_walki_i_duchowa_energia Dragonborn_Monk
6 Dragonborn Straznik_Swiatla Dragonborn_Paladin
7 Dragonborn Infiltracja_i_Obrazenia_krytyczne Dragonborn_Rouge
8 Dragonborn Wsparcie_i_Leczenie Dragonborn_Cleric
9 Dragonborn Inzynieria_magiczna Dragonborn_Artificer
10 Dragonborn Wiedza_i_Arkana_Magi Dragonborn_Wizard
11 Dragonborn Narodzony_z_Magia Dragonborn_Sourcerer
12 Dragonborn Muzyka_i_Magia Dragonborn_Bard
13 Dragonborn Zrecznosc_i_Bron_dystansowa Dragonborn_Ranger
14 Dragonborn Magia_paktow_i_Cienie Dragonborn_Warlock
15 Dragonborn Przywiazanie_do_natury Dragonborn_Druid
16 Dwarf Furia_i_Wscieklosc Dwarf_Barbarian
17 Dwarf Mistrz_Miecza Dwarf_Fighter
18 Dwarf Sztuki_walki_i_duchowa_energia Dwarf_Monk
19 Dwarf Straznik_Swiatla Dwarf_Paladin
20 Dwarf Infiltracja_i_Obrazenia_krytyczne Dwarf_Rogue
21 Dwarf Wsparcie_i_Leczenie Dwarf_Cleric
22 Dwarf Inzynieria_magiczna Dwarf_Artificer
23 Dwarf Wiedza_i_Arkana_Magi Dwarf_Wizard
24 Dwarf Narodzony_z_Magia Dwarf_Sorcerer
25 Dwarf Muzyka_i_Magia Dwarf_Bard
26 Dwarf Zrecznosc_i_Bron_dystansowa Dwarf_Ranger
27 Dwarf Magia_paktow_i_Cienie Dwarf_Warlock
28 Dwarf Przywiazanie_do_natury Dwarf_Druid
29 Elf Furia_i_Wscieklosc Elf_Barbarian
30 Elf Mistrz_Miecza Elf_Fighter
31 Elf Sztuki_walki_i_duchowa_energia Elf_Monk
32 Elf Straznik_Swiatla Elf_Paladin
33 Elf Infiltracja_i_Obrazenia_krytyczne Elf_Rogue
34 Elf Wsparcie_i_Leczenie Elf_Cleric
35 Elf Inzynieria_magiczna Elf_Artificer
36 Elf Wiedza_i_Arkana_Magi Elf_Wizard
```

Drzewo wygenerowane przez program:



8. Implementacja systemu ekspertowego w języku Python (automatyczny)

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import LabelEncoder
import tkinter as tk
from tkinter import ttk

def get_class_description(file_path, selected_class):
    try:
        with open(file_path, 'r', encoding='utf-8') as file:
            data = file.read().split('\n\n')
            for section in data:
                if selected_class in section:
                    return section
            return "Description not found for this class."
    except FileNotFoundError:
        return "File not found."

def get_race_description(file_path, selected_race):
    try:
        with open(file_path, 'r', encoding='utf-8') as file:
            data = file.read().split('\n\n')
            for section in data:
                if selected_race in section:
                    return section
            return "Description not found for this race."
    except FileNotFoundError:
        return "File not found."

def get_main_description(file_path, selected_race):
    try:
        with open(file_path, 'r', encoding='utf-8') as file:
            data = file.read().split('\n\n')
            for section in data:
                if selected_race in section:
                    return section
            return "Description not found for this race."
    except FileNotFoundError:
        return "File not found."

def predict_character():
    selected_race = race_combobox.get()
    selected_class = class_combobox.get()

    if selected_race in race_map and selected_class in class_map:
        race_label_enc = race_map[selected_race]
        class_label_enc = class_map[selected_class]

        selected_race_enc = race_encoder.transform([selected_race])[0]
        selected_class_enc = class_encoder.transform([selected_class])[0]

        prediction = dtree.predict([selected_race_enc, selected_class_enc])

        if prediction:
            predicted_character = prediction[0]
            result_label.config(text=f"Wybrana postać: {predicted_character}")

            main_description = get_main_description('main_descriptions.txt', predicted_character)
            map_description = main_description.split(':', 1)[1].strip() if ':' in main_description else main_description
            result_description_label.config(text=main_description)
        else:
            result_label.config(text="Prediction not found in map.")
    else:
        result_label.config(text="Please select a valid race and class.")

df = pd.read_csv("csv.csv", delimiter=';')

race_encoder = LabelEncoder()
class_encoder = LabelEncoder()

df['wybierz_race_enc'] = race_encoder.fit_transform(df['wybierz_race'])
df['wybierz_klasa_enc'] = class_encoder.fit_transform(df['wybierz_klasa'])

race_map = dict(zip(df['wybierz_race'], df['wybierz_race_enc']))
class_map = dict(zip(df['wybierz_klasa'], df['wybierz_klasa_enc']))
inverse_race_map = {v: k for k, v in race_map.items()}

features = ['wybierz_race_enc', 'wybierz_klasa_enc']
X = df[features]
y = df['postac']

dtree = DecisionTreeClassifier()
dtree.fit(X, y)

root = tk.Tk()
root.title("Przewidywanie Postaci")

main_frame = tk.Frame(root)
main_frame.pack(padx=20, pady=20)

race_frame = tk.Frame(main_frame)
race_frame.pack()

race_label = tk.Label(race_frame, text="Wybierz race:")
race_label.pack(side=tk.LEFT, padx=10)

races = list(race_map.keys())
race_combobox = tk.Combobox(race_frame, values=races, width=50)
race_combobox.pack(side=tk.LEFT)

race_description_label = tk.Label(main_frame, text="", wraplength=400)
race_description_label.pack(pady=5)

class_frame = tk.Frame(main_frame)
class_frame.pack()

class_label = tk.Label(class_frame, text="Wybierz klasę:")
class_label.pack(side=tk.LEFT, padx=10)

classes = list(class_map.keys())
class_combobox = tk.Combobox(class_frame, values=classes, width=30)
class_combobox.pack(side=tk.LEFT)

class_description_label = tk.Label(main_frame, text="", wraplength=400)
class_description_label.pack(pady=5)

predict_button = tk.Button(main_frame, text="Przewidywany wynik", command=predict_character)
predict_button.pack(pady=10)

result_label = tk.Label(main_frame, text="", font=("Arial", 12))
result_label.pack()

result_description_label = tk.Label(main_frame, text="", wraplength=400)
result_description_label.pack(pady=5)

def update_race_description(event):
    selected_race = race_combobox.get()
    race_description = get_race_description('race_descriptions.txt', selected_race)
    race_description = race_description.split(':', 1)[1].strip() if ':' in race_description else race_description
    race_description_label.config(text=race_description)

race_combobox.bind("<<ComboboxSelected>>", update_race_description)

def update_class_description(event):
    selected_class = class_combobox.get()
    class_description = get_class_description('class_descriptions.txt', selected_class)
    class_description = class_description.split(':', 1)[1].strip() if ':' in class_description else class_description
    class_description_label.config(text=class_description)

class_combobox.bind("<<ComboboxSelected>>", update_class_description)

root.mainloop()
```


Powyższy kod stanowi implementację systemu eksperckiego w języku Python, służącego do prognozowania postaci w świecie D&D na podstawie wybranych rasy i klasy. Poniżej przedstawiono kluczowe elementy oraz działanie kodu:

Biblioteki:

- Użyto bibliotek takich jak **pandas** do manipulacji danymi, **DecisionTreeClassifier** z **scikit-learn** do budowy drzewa decyzyjnego oraz **tkinter** do stworzenia interfejsu graficznego.

Wczytanie Danych:

- Dane postaci są wczytywane z pliku CSV o nazwie "csv.csv" z wykorzystaniem biblioteki **pandas**.

Label Encoding:

- Wykorzystano **LabelEncoder** z **scikit-learn** do zakodowania etykiet (rasy i klasy postaci) na liczby całkowite, co jest wymagane do zastosowania drzewa decyzyjnego.

Budowa Drzewa Decyzyjnego:

- Stworzono drzewo decyzyjne (**DecisionTreeClassifier**), które przewiduje postać na podstawie wybranych rasy i klasy.

Interfejs Graficzny:

- Stworzono prosty interfejs graficzny przy użyciu **tkinter** z elementami takimi jak etykiety, pola wyboru (**Combobox**), przyciski oraz pola wyświetlające wyniki.

Przewidywanie Postaci:

- Funkcja **predict_character** korzysta z drzewa decyzyjnego do prognozowania postaci na podstawie wybranych rasy i klasy. Wyniki są prezentowane w etykietach na interfejsie.

Aktualizacja Opisów:

- Dodano funkcje aktualizujące opisy ras i klas w odpowiedzi na wybór użytkownika.

Ten kod umożliwia interaktywne tworzenie postaci w świecie D&D, a interfejs graficzny ułatwia użytkownikom wybór rasy i klasy, prezentując jednocześnie prognozowaną postać w opisie.

Wynik działania programu:

Przewidywanie Postaci

Wybierz rasę:

Dwarf

-Wygląd: Dwarfy to krótkowzrostłe i krępe istoty o ciemnej skórze, brodate twarze, często ozdobione metalowymi ozdobami. Są znani ze swojej wytrzymałości i siły.

-Cechy: Dwarfy są utalentowane w rzemiośle, górnictwie i wojennej sztuce. Są też znani ze swojego uporu i lojalności.

-Historia: Dwarfy są znane z budowy imponujących miast pod ziemią i skomplikowanego systemu klanów. Są odwiecznymi wrogami orków i goblinów.

Wybierz Klasę:

Sztuki_walki_i_duchowa_energia

-Mnisi to mistrzowie sztuk walki, którzy wykorzystują swój własny ciało i duchową energię do walki.

Ta klasa skupia się na szybkości, zwinności i wykorzystywaniu unikalnych zdolności, takich jak techniki ciosów krytycznych i obrona przed magią.

Mnisi często żyją w klasztorach i dążą do duchowego doskonalenia.

Przewidywany wynik

Wybrana postać: Dwarf_Monk

Stworzona Postać!

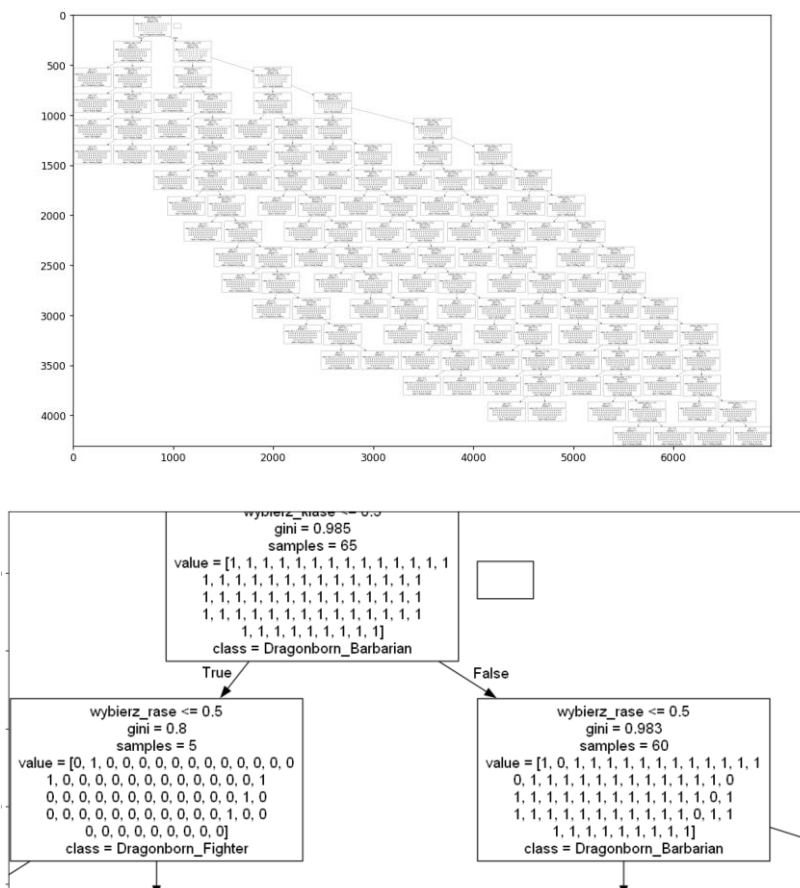
Imię: Nalda Ciepłozębna

Rasa: Dwarf

Klasa: Monk

Historia: Nalda to mniszka, która opuściła klasztor, aby doskonalić umiejętności walki i ducha.

Wygląd mydecisiontree.



[illegible]

Powyższy kod to ręczna implementacja systemu eksperckiego w języku Python z wykorzystaniem biblioteki PyQt do prognozowania postaci w świecie D&D na podstawie wybranych rasy i klasy. Kod został skrócony aby umożliwić wstawienie go do dokumentacji jego pełna wersja znajduje się w plikach. Poniżej przedstawiono kluczowe elementy oraz działanie kodu:

Biblioteki i Interfejs Graficzny:

- Kod wykorzystuje bibliotekę PyQt5 do stworzenia interfejsu graficznego, umożliwiającego użytkownikowi wybór rasy i klasy postaci w grze D&D. Widżety, takie jak etykiety, pola wyboru (**QComboBox**), przyciski oraz pola wyświetlające wyniki, są używane do zbudowania intuicyjnego interfejsu.

Wczytywanie Opisów:

- Funkcja **load_descriptions** odpowiedzialna jest za wczytywanie opisów ras i klas postaci z plików tekstowych. To pozwala na dynamiczne aktualizowanie interfejsu z nowymi danymi.

Reguły Systemu Eksperckiego:

- Implementacja reguł systemu eksperckiego oparta jest na bibliotece **durable_rules**. Reguły te definiują związki między wybraną rasą a klasą postaci. Każda reguła zawiera opis postaci oraz dodatkowe informacje, co pozwala na elastyczne dodawanie nowych postaci do systemu.

Prezentacja Wyników:

- Po dokonaniu wyboru rasy i klasy przez użytkownika, system ekspercki przewiduje postać na podstawie zdefiniowanych reguł. Opis wybranej postaci prezentowany jest w interfejsie użytkownika, co umożliwia szybkie zrozumienie, kim jest stworzona postać.

Zakończenie Programu:

- Interfejs zawiera przycisk "Zakończ", który umożliwia użytkownikowi zakończenie programu. Ta funkcjonalność pozwala na wygodne zamykanie aplikacji po utworzeniu postaci.

Działanie Reguł:

- System ekspercki działa na zasadzie wnioskowania opartego na regułach. Po dokonaniu wyboru przez użytkownika, system przegląda zdefiniowane reguły, aby znaleźć odpowiednią postać. Wynik wnioskowania prezentowany jest w czytelny sposób na interfejsie.

Interaktywność:

- Kod uwzględnia interakcję z użytkownikiem poprzez obsługę zdarzeń, takich jak wybór rasy i klasy. Działa to jak interaktywny kreator postaci, który dostarcza informacji o każdej postaci oraz jej historii.

Wynik działania programu:

Character Creator

Choose Race:

Dragonborn

-Wygląd: Dragonborn są ludźmi ze skórą w różnych odcieniach kolorów smoków, jak czerwony, niebieski, zielony lub czarny. Mają rogopodobne wyrostki na głowach oraz skrzydlate narośla na plecach.
-Cechy: Dragonborn są znani ze swojej siły i dumy. Każda podrasa Dragonborn ma własne zdolności związane z rodzajem smoka, którego przodkowie wywodzą się od.
-Historia: Dragonborn wywodzą się od smoków i są dumnie związani ze swoim pochodzeniem. Mają bogatą historię związaną z konfliktami i współpracą z innymi rasami w świecie D&D.

Choose Class:

Infiltracja i Obrazenia krytyczne

Rogue to mistrzowie skradania się, obserwacji i podstępów. Często pracują jako złodzieje, szpiegi lub zabójcy.
Ta klasa skupia się na unikaniu walki, skradaniu się i zadawaniu krytycznych obrażeń wrogom. Rogues są biegli w rozbijaniu pułapek, otwieraniu zamków i wykrywaniu ukrytych zagrożeń.

Create Character

Character Creator

Race:

-Wygląd: Dragonborn są ludźmi ze skórą w różnych odcieniach kolorów smoków, jak czerwony, niebieski, zielony lub czarny. Mają rogopodobne wyrostki na głowach oraz skrzydlate narośla na plecach.
-Cechy: Dragonborn są znani ze swojej siły i dumy. Każda podrasa Dragonborn ma własne zdolności związane z rodzajem smoka, którego przodkowie wywodzą się od.
-Historia: Dragonborn wywodzą się od smoków i są dumnie związani ze swoim pochodzeniem. Mają bogatą historię związaną z konfliktami i współpracą z innymi rasami w świecie D&D.

Class:

Rogue to mistrzowie skradania się, obserwacji i podstępów. Często pracują jako złodzieje, szpiegi lub zabójcy.
Ta klasa skupia się na unikaniu walki, skradaniu się i zadawaniu krytycznych obrażeń wrogom. Rogues są biegli w rozbijaniu pułapek, otwieraniu zamków i wykrywaniu ukrytych zagrożeń.

Character Information:

Postać: Dragonborn Cleric

Opis postaci:

Imię: Tordek Kamienne Serce
Rasa: Dwarf
Klasa: Rogue
Historia: Tordek jest zręcznym złodziejem i przemytnikiem, który opuścił góry, aby prowadzić życie w podziemnym świecie intryg i skradania.

Zakończ

10. Porównanie implementacji SPHINX i Python

Według mojej oceny, Python jest znacznie lepszym wyborem w porównaniu z Sphinxem. Python oferuje prostotę, czytelność kodu i ogromną społeczność dzięki czemu jest łatwiejszy do nauki. Jest bardziej wszechstronny co sprawdza się w większych projektach. Ponad to jest popularny i posiada wiele bibliotek ułatwiających pracę przy implementacji naszego rozwiązania.