

Skyrim Search App – Dokumentacja

Cele projektu:

Celem projektu było stworzenie aplikacji desktopowej pozwalającej wyszukiwać potrzebne graczowi przedmioty w grze Skyrim, w zależności od dostępnego budżetu, siły rażenia, wagi, itp.

Wykorzystane technologie:

Do realizacji projektu zdecydowano się użyć dwóch technologii: do stworzenia interfejsu użytkownika wykorzystano język programowania Python, a do strony backendowej – bazę danych MongoDB. Wybrano taką bazę, ponieważ była ona doskonała do postawionych celów – wystarczyło stworzyć kolekcje podobnych dokumentów, odwzorowujących klasę abstrakcji przedmiotów w grze.

Pliki i ich zawartość:

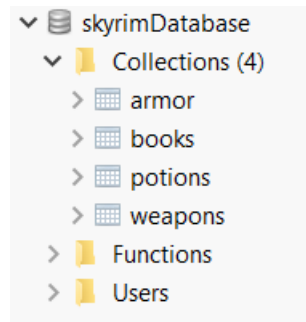
1. Plik constants.py – zawiera stałe, które są używane przez inne pliki, takie jak czcionki, używane kolory oraz słownik tłumaczenia pól bazy danych do wyświetlania w GUI.
2. Plik database_functions.py – zawiera funkcje bazy danych, pozwalające na edytowanie, dodawanie i usuwanie rekordów bazy.
3. Plik exceptions.py – zawiera zdefiniowany wyjątek, który używany jest w innych częściach projektu.
4. Plik helping_functions.py – zawiera dodatkowe funkcje używane na przykład podczas ewaluacji zmiennych (typ float/int).
5. Plik logo.png – obrazek gry Skyrim, wyświetlający się w oknie aplikacji.
6. Plik main_project_file.py – łączy w sobie wszystkie funkcjonalności, tworzy okno aplikacji, pobiera dane od użytkownika.
7. Plik requirements.txt – pozwala na zainstalowanie potrzebnych Pythonowych modułów.

Uruchomienie aplikacji:

Aby uruchomić aplikację ze standardową bazą danych, wystarczy pobrać pliki z repozytorium, zainstalować wymagane ustawienia z pliku requirements.txt. Następnie aplikację można uruchomić na dwa sposoby – albo przez Pycharma, jako skrypt uruchomieniowy wybierając main_project_file, albo poprzez Command Line: przejść do folderu z projektem (za pomocą polecenia cd), następnie wpisać polecenie: python main_project_file.py.

Baza danych – backend dla przedmiotów:

Aby ułatwić pracę z projektem, zdecydowano się na bazę danych w chmurze. Pozwoliło to nie dodawać dwukrotnie tych samych dokumentów do baz lokalnych.



Rysunek 1 - Zbiór kolekcji bazy danych

Jak widać na Rysunku 1, baza danych składa się z czterech kolekcji, obrazujących klasy abstrakcji przedmiotów dostępnych w grze: zbroję, książki, eliksiry oraz broń.



Rysunek 2 - Przykładowy dokument kolekcji armor

Rysunek 2 przedstawia przykładowy dokument z kolekcji armor. Pola dokumentu odpowiadają informacjom, jakie można uzyskać prosto z gry o danym przedmiocie (pomijają natomiast takie informacje, jak id danego przedmiotu, gdyż do rozgrywki bez użycia kodów nie jest to potrzebne). Zdecydowano się także na pole tablicowe enchantments, które zawiera wszystkie zaklęcia narzucone na dany przedmiot. Podjęto takie rozwiązanie, ponieważ specyfika gry pozwala przedmiotowi na bycie zaklętym 0, 1, 2 lub więcej zaklęciami.

▼ (1) ObjectId("5e9c5ace5a045cabeca6c06e")	{ 6 fields }
_id	ObjectId("5e9c5ace5a045cabeca6c06e")
name	Spokój
level	Uczeń
school_of_magic	Iluzja
mana_cost	153.0
description	Istoty i ludzie na maksymalnie 9 poziomie uciekają od walki przez 30 s.

Rysunek 3 - Przykładowy dokument kolekcji books

Rysunek 3 pokazuje przykładowy dokument z kolekcji books. Księgi, które spotyka się w grze, to księgi czarów o takiej samej wadze, dlatego też zdecydowano się nie dodawać tej informacji do bazy danych.

▼ (1) ObjectId("5e9af7c38515456bc2857c36")	{ 6 fields }
_id	ObjectId("5e9af7c38515456bc2857c36")
name	Misktura pomniejszego uleczenia
type_of_restoring	health
weight	0.5
value	17.0
effect	Przywraca 25 punktów zdrowia

Rysunek 4 - Przykładowy dokument kolekcji potions

▼ (2) ObjectId("5e9c5bdc5a045cabeca6e254")	{ 9 fields }
_id	ObjectId("5e9c5bdc5a045cabeca6e254")
name	Ebonowy miecz
basic_attack	13.0
weight	15.0
value	720.0
is_unique	false
enchantment	[0 elements]
is_onehanded	true
type	Miecz

Rysunek 5 - Przykładowy dokument kolekcji weapons

Pozostałe dwa rysunki pokazują pozostałe dwie kolekcje bazy danych.

Pythonowe pliki – interfejs użytkownika:

Do wykonania interfejsu użytkownika zdecydowano się wykorzystać bibliotekę Tkinter. Jest to znana biblioteka, do której można znaleźć także dużą ilość tutoriali oraz pytań na StackOverflow. Do tego łatwość jej użytkowania zadziałała na plus, więc zdecydowano się na tę bibliotekę. Do komunikacji z bazą danych wykorzystano pymongo, z którego zaimportowano MongoClient.

Plik constants.py – tam, gdzie trzymamy stałe:

```
1  bg_color = "#717171"
2  fg_color = "#eeeeee"
3  font = ("Courier", 20)
4  small_font = ("Courier", 14)
5
6  # słownik przekształcający nazwy pól kolekcji do postaci czytelnej dla człowieka
7  labels = {
8      "name": "Nazwa",
9      "basic_attack": "Podstawowy atak",
10     "weight": "Waga",
11     "value": "Wartość",
12     "is_unique": "Czy unikalny",
13     "enchancements": "Efekty",
14     "enchantment": "Efekty",
15     "type": "Typ",
16     "is_onehanded": "Czy jednoręczny",
17     "basic_armor": "Podstawowa wartość zbroi",
18     "is_light": "Czy lekka",
19     "type_of_restoring": "Przywraca",
20     "effect": "Efekt",
21     "description": "Opis",
22     "level": "Poziom",
23     "mana_cost": "Koszt Many",
24     "school_of_magic": "Szkoła"
25 }
```

Plik constants.py zawiera stałe, które są używane do późniejszej implementacji projektu. Są to czcionki oraz ich rozmiary, kolory, jakie są używane oraz słownik labels przekształcający nazwy pól dokumentów do postaci, która jest przyswajalna dla typowego użytkownika.

Plik database_functions.py – zabawa z bazą danych:

Plik database_functions.py odpowiada za połączenie oraz interakcję z bazą danych.

Połączenie z bazą odbywa się za pomocą pierwszych linijek kodu:

```
1 from pymongo import MongoClient
2
3 client = MongoClient('mongodb+srv://dawid:dawid99@skyrimcluster-j4jje.gcp.mongodb.net/test')
4 db = client.skyrimDatabase
5
```

Aby połączyć się ze swoją bazą danych, wystarczy wkleić odpowiedni link z Atlasa w miejsce `client = MongoClient('<<Twój link>>')`. Żadne dane przykładowe nie są ładowane do bazy w momencie uruchomienia aplikacji, więc jeżeli wpisze się inną bazę, będzie ona pusta!

Znajdowanie przedmiotów:

Następnie używane są funkcje znajdujące przedmioty spełniające określone wymagania:

```
6 #finding functions
7 def findWeapon(min_s, max_s, min_v, max_v, min_w, max_w):
8     return db.weapons.find({
9         "basic_attack": {"$gte": min_s, "$lte": max_s},
10        "value": {"$gte": min_v, "$lte": max_v},
11        "weight": {"$gte": min_w, "$lte": max_w},
12    })
13
```

Funkcja `findWeapon` pozwala na znalezienie przedmiotów z kolekcji `weapons` i zwraca te przedmioty. Funkcja `findArmor` działa podobnie, ale wyszukuje przedmioty z kolekcji `armor`.

```
def findArmor(min_s, max_s, min_v, max_v, min_w, max_w):
    return db.armor.find({
        "basic_armor": {"$gte": min_s, "$lte": max_s},
        "value": {"$gte": min_v, "$lte": max_v},
        "weight": {"$gte": min_w, "$lte": max_w},
    })
```

```
def findBook(min_m, max_m, spell_school, spell_level):
    if spell_school == "" and spell_level == "":
        return db.books.find({
            "mana_cost": {"$gte": min_m, "$lte": max_m}
        })
    elif spell_school == "":
        return db.books.find({
            "level": spell_level,
            "mana_cost": {"$gt": min_m, "$lt": max_m}
        })
    elif spell_level == "":
        return db.books.find({
            "school_of_magic": spell_school,
            "mana_cost": {"$gt": min_m, "$lt": max_m}
        })
    else:
        return db.books.find({
            "level": spell_level,
            "school_of_magic": spell_school,
            "mana_cost": {"$gt": min_m, "$lt": max_m}
        })
```

Mimo swojej rozbudowanej postaci, funkcja findBook działa w ten sam sposób, rozpatrzone zostały jednak przypadki skrajne, polegające na niepodaniu przez użytkownika jednej z wartości potrzebnych do znalezienia przedmiotu.

```
def findPotion(min_v, max_v, search):
    if search == "":
        return db.potions.find({
            "value": {"$gt": min_v, "$lt": max_v}
        })
    return db.potions.find({
        "type_of_restoring": search,
        "value": {"$gt": min_v, "$lt": max_v}
    })
```

Funkcja findPotion znajduje eliksiry z takim samym rozpatrzeniem przypadków skrajnych, których jest jednak nieco mniej.

Dodawanie przedmiotów do bazy danych:

Aby użytkownikowi wygodnie było dodać przedmiot, zapewniono także odpowiednie funkcje komunikacji z bazą danych. Ogólna ich zasada działania to stworzenie słownika <<pole z bazy danych>> : <<wartość podana przez użytkownika>>. Później słownik taki wstawia się do odpowiedniej kolekcji.

```
def add_weapon(name, stat, value, weight, type, is_unique, is_onehanded, enchant1, enchant2):  
    dict = {  
        "name": name,  
        "basic_attack": stat,  
        "weight": weight,  
        "value": value,  
        "type": type,  
        "is_unique": is_unique,  
        "is_onehanded": is_onehanded,  
        "enchantment": [enchant1, enchant2],  
    }  
    db.weapons.insert_one(dict)  
    return 1
```

Zrzut ekranu pokazuje przykładowe dodawanie przedmiotu do bazy danych.

Modyfikowanie przedmiotów z bazy:

```
def modify_weapon(id, name, stat, value, weight, type, is_unique, is_onehanded, enchant1, enchant2):  
    my_query = {"_id": id}  
    new_values = {"$set": {  
        "name": name,  
        "basic_attack": stat,  
        "weight": weight,  
        "value": value,  
        "type": type,  
        "is_unique": is_unique,  
        "is_onehanded": is_onehanded,  
        "enchantment": [enchant1, enchant2],  
    }}  
    db.weapons.update_one(my_query, new_values)  
    return 1
```


Usuwanie przedmiotów z bazy:

```
def delete_weapon(weapon):  
    my_query = {"_id": weapon["_id"]}   
    db.weapons.delete_one(my_query)  
    return 1
```

Wygląd aplikacji:




Główne okno aplikacji pokazuje interfejs, dzięki któremu można przejść do wyszukiwania bądź dodawania przedmiotów. Poniżej przedstawione są widoki z poszczególnych wyborów:

Skyrim app

The Elder Scrolls V

SKYRIM



Min attack

Max attack

☐ Don't care

Min value

Max value

☐ Don't care

Min weight

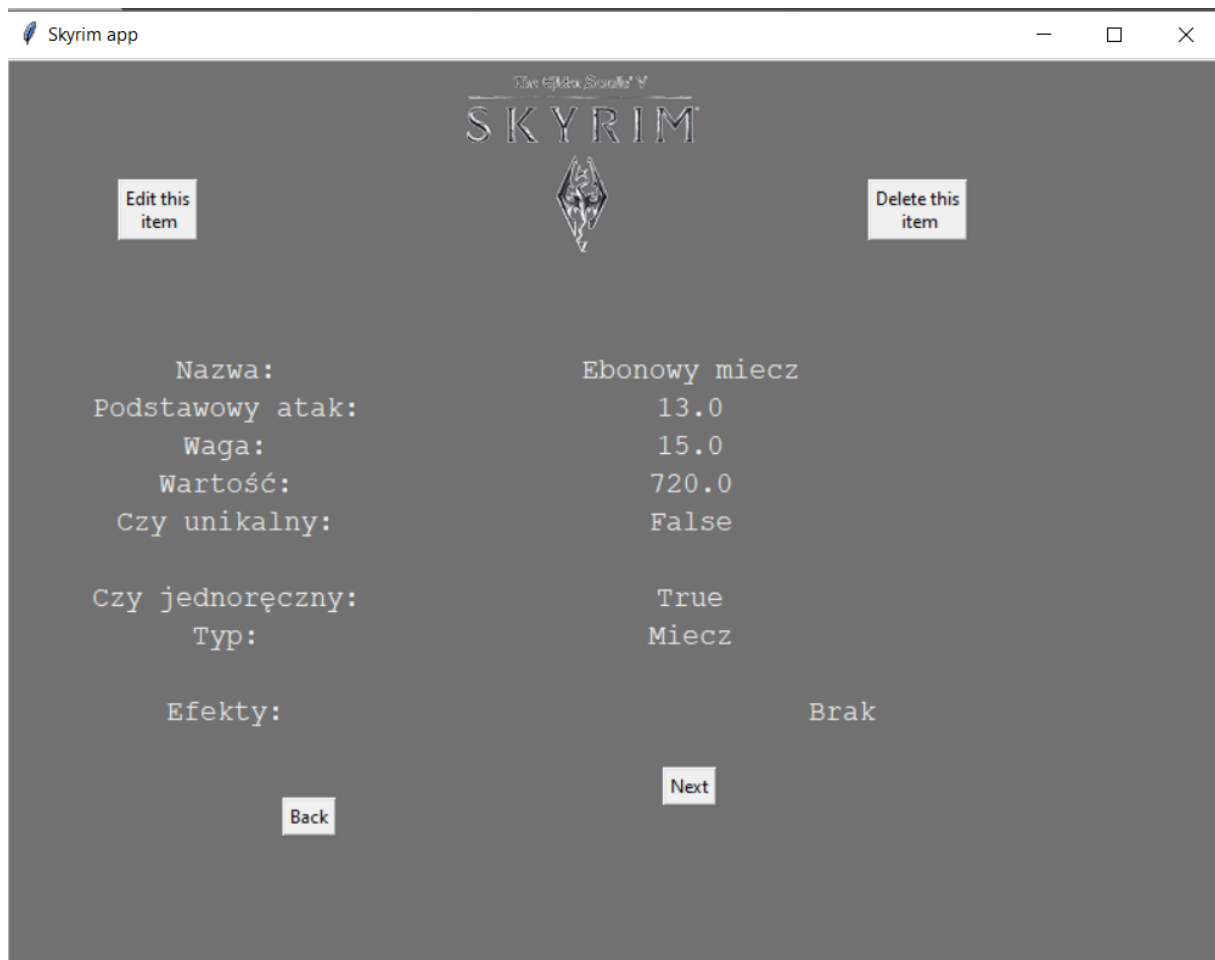
Max weight

☐ Don't care

Back

Search!

Ekran wyszukiwania przedmiotów. Po kliknięciu Search!, wyświetla się poniższe okno, pozwalające na przejrzanie przedmiotów spełniających podane przez użytkownika warunki:




Po użyciu przycisku Next, przegląda się dalsze przedmioty pasujące do warunków postawionych przez użytkownika. Przycisk Delete this item pozwala na usunięcie danego przedmiotu z bazy danych. Przycisk Edit this item pozwala na zmodyfikowanie danego przedmiotu.

Skyrim app

The Elder Scrolls V

SKYRIM



Atak:

Typ:

Nazwa:

Zaklęcie 1:

Waga:

Zaklęcie 2:

Wartość:

Dodaj

Czy jest unikalny?

Back

Czy jest jednoręczny?

Dany widok pokazuje interfejs aplikacji do dodawania przedmiotów do bazy danych. Pola Atak, Waga i Wartość muszą być numeryczne, inaczej po kliknięciu przycisku Dodaj aplikacja przesunie użytkownika do danego pola, aby wypełnił je poprawnie.

a