



**Politechnika  
Śląska**

Dokumentacja projektu

2019/2020

**POiG**

*„Dungeons & Dragons”*

Kierunek: Informatyka

Członkowie zespołu:

*Filip Gabryszewski*

*Piotr Hadam*

Gliwice, 2019/2020

# 1 Wprowadzenie i Opis

Na chwilę obecną nie ma wystarczająco aplikacji wspomagających w dostatecznie dużym stopniu zautomatyzowanie procesów w grach turowych typu D&D. Z racji tego postanowiliśmy stworzyć bazę zawierającą wiele podstawowych funkcji związanych z tworzeniem i przechowywaniem danych związanych z D&D 5e, która podejmuje próbę rozwiązania niektórych problemów pojawiających się przy np. tworzeniu karty postaci i umożliwia automatyzację części procesów.

## 2 Wymagania

Baza danych powinna zawierać wszystkie możliwe i niezbędne informacje na temat danej postaci (charakteru), którego kartę postaci mamy zamiar stworzyć i informacje na temat jej cech. Aplikacja powinna odczytywać je wszystkie i wyprowadzać resztę istotnych elementów ułatwiających rozgrywkę i umożliwiać obsługę elementów zawartych w bazie.

The image shows a character sheet template for Dungeons & Dragons 5e. The sheet is divided into several sections:

- Header:** DUNGEONS & DRAGONS logo, CHARACTER NAME, CLASS & LEVEL, ALIGNMENT, RACE, SEX, DESCRIPTION.
- Ability Scores:** STRENGTH, DEXTERITY, CONSTITUTION, INTELLIGENCE, WISDOM, CHARISMA. Each score has a corresponding skill list.
- Hit Points:** HIT POINTS, ARMOR CLASS, PROFICIENCY BONUS, CURRENT HIT POINTS.
- Skills:** A list of skills with checkboxes: Acrobatics (Dex), Animal Handling (Wis), Arcana (Int), Athletics (Str), Deception (Cha), History (Int), Insight (Wis), Intimidation (Cha), Investigation (Int), Medicine (Wis), Nature (Int), Perception (Wis), Performance (Cha), Persuasion (Cha), Religion (Int), Sleight of Hand (Dex), Stealth (Dex), Survival (Wis).
- Combat:** SPEED, INITIATIVE, ARMOR, SHIELD, WEAPONS & ATTACKS, DMG, TYPE, AMMO.
- Other:** SAVING THROWS, PASSIVE WISDOM (PERCEPTION), PROFICIENCIES, TRAITS & ABILITIES, LANGUAGES.

Fragment karty postaci

W bazie musimy zatem zawrzeć wszystkie "dane osobowe" postaci takie jak imię, poziom, klasę czy informacje na temat zdolności i umiejętności jak i zarazem na temat posiadanych różnego rodzaju przedmiotów takich jak broń czy zbroje albo po prostu zwykłe przedmioty w inwertarzu. Baza danych może również przechowywać ścieżki do obrazu jednak z powodu braku serwera z przestrzenią dyskową raczej nie korzystamy z tej opcji. Ogólne cele:

- Dostarczanie informacji o postaci takich jak imię i zdolności,
- Dostarczanie listy broni postaci
- Dostarczanie listy zbroi i przedmiotów
- Dostarczanie listy zaklęć umianej przez postać
- Dostarczanie listy wszystkich dostępnych zaklęć

### 3 Przebieg realizacji

Aplikacja jest napisana przy użyciu MVVM. Struktura plików prezentuje się następująco:

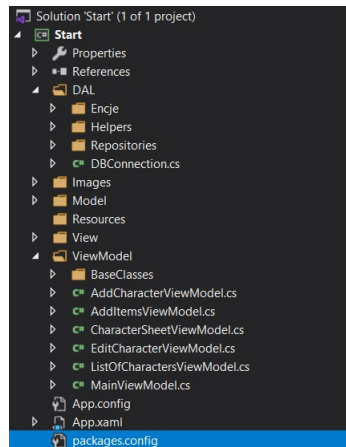


Diagram związków encji

#### 3.1 Podział ról i wykonanie zadań

Piotr Hadam

- Zaimplementowanie warstwy DAL
- Częściowe wypełnienie bazy danych
- Zaprojektowanie ogólnego modelu aplikacji i implementacja większości funkcji

Filip Gabryszewski

- Zaprojektowanie bazy danych i częściowe uzupełnienie baz danych
- Zaprojektowanie widoku karty postaci
- Implementacja pojedynczych funkcji dla aplikacji

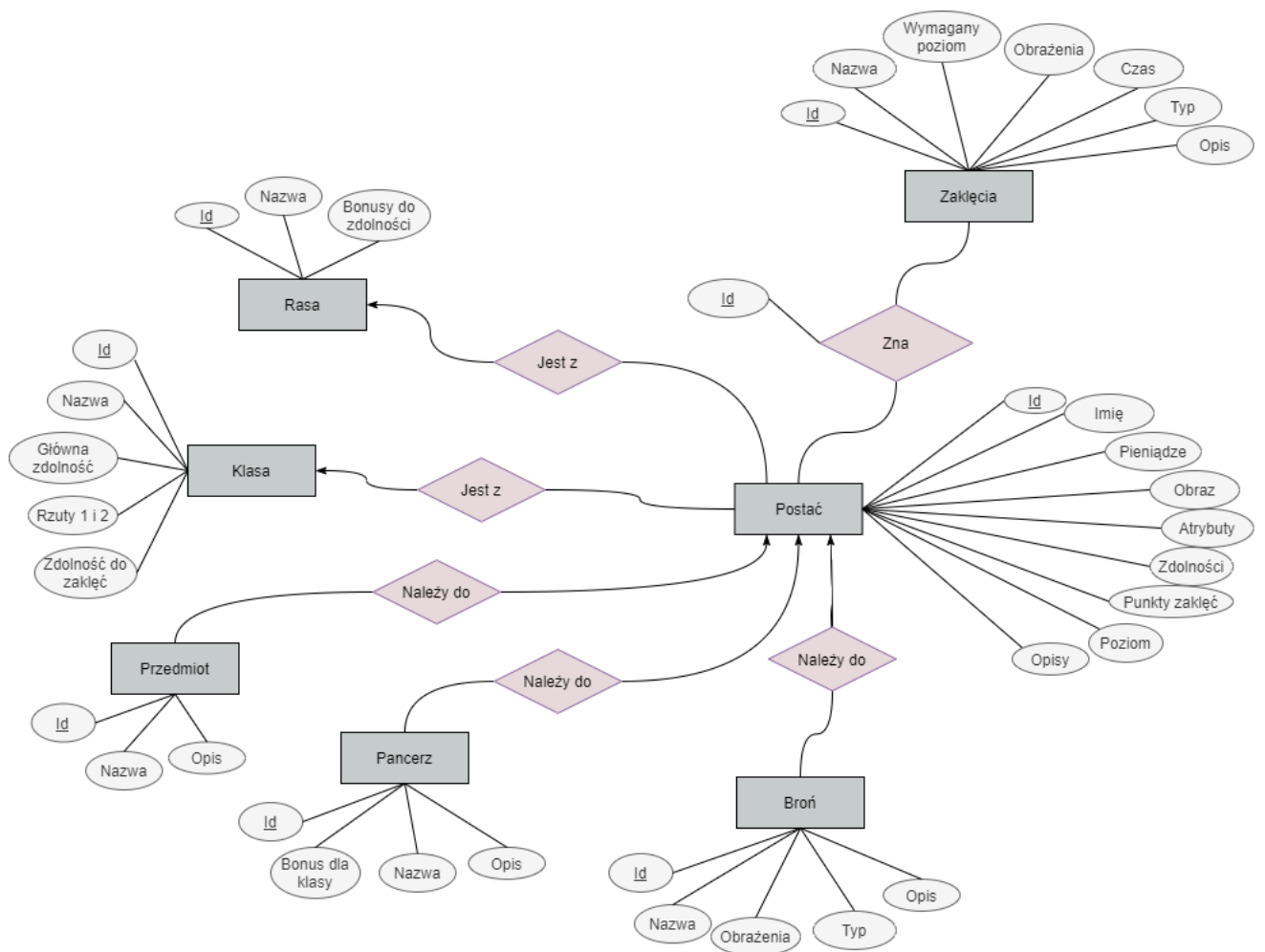
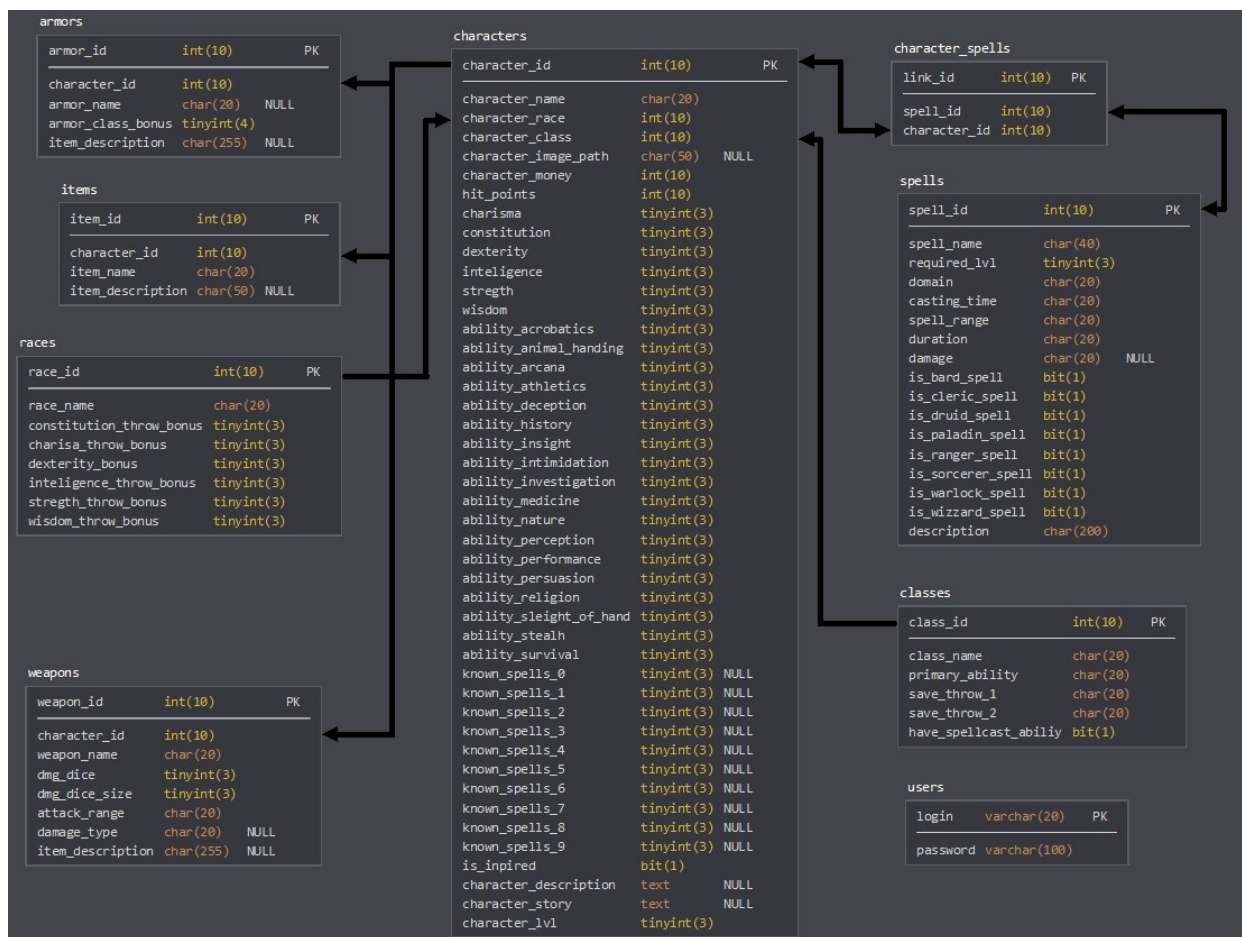


Diagram związków encji



Model relacyjny

Baza danych połączona jest z aplikacją okienkową napisaną w języku C#. Połączone są one poprzez warstwę dostępu do danych DAL. Każda encja bazy ma swoje odwzorowanie jako klasa. Każda klasa posiada konstruktor pobierający dane z bazy. Dodatkowo zostały stworzone klasy repozytoriów - to dzięki nim aplikacja ma dostęp do danych. Wszystkie repozytoria zawierają metody pobierające kolejne rekordy tabeli i tworzące nowe obiekty. Ponadto repozytoria odpowiadające encjom otwartym dla użytkowników, tzn. „characters”, „items”, „armors”, „weapons” oraz „character\_spells” (encja łącząca postacie z zaklęciami) posiadają metody odpowiedzialne za dopisywanie do tablic kolejnych rekordów na podstawie utworzonych w aplikacji obiektów, edytowanie już istniejących rekordów oraz usuwanie ich.

Jako przykład kodu klasę odpowiedzialną za połączenie z określoną bazą danych „DBConnection”:

```
1 class DBConnection
2 {
3     private static MySqlConnectionStringBuilder stringBuilder;
4
5     public static string Nickname { get; private set; }
6     private static string Password { get; set; }
7     public static string Server { get; set; }
8     private static string Database { get; set; }
9     private static uint Port { get; set; }
10
11     public static DBConnection Instance
12     {
13         get => new DBConnection();
14     }
15
16     public MySqlConnection Connection =>
17     new MySqlConnection(stringBuilder.ToString());
18
19     private DBConnection()
20     {
21         stringBuilder = new MySqlConnectionStringBuilder
22         {
23             UserID = Nickname,
24             Password = Password,
25             Server = Server,
26             Database = Database,
27             Port = Port,
28             CharSet = "utf8"
29         };
30     }
31
32     public static void Login(string user, string password)
33     {
34         Nickname = user;
35         Password = password;
36     }
37
38     public static bool LoginAsRoot()
39     {
40         try
41         {
42             Nickname = "root";
43             Password = "";
44             Server = "localhost";
45             Database = "dnd_characters";
46             Port = 3306;
47             return true;
48         }
49         catch { return false; }
50     }
51 }
```

---

## 4 Instrukcja Obsługi

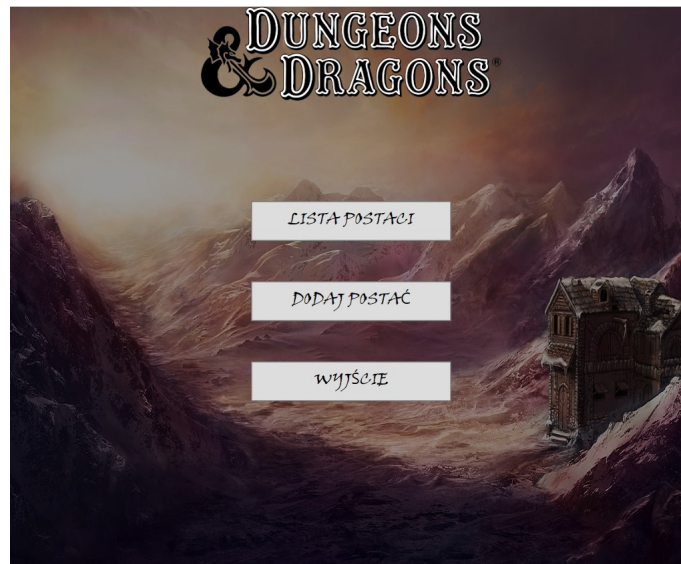
### 4.1 Użytkownik

Użytkownik również może dostać się do bazy poprzez konsolę po zalogowaniu się swoimi danymi - nazwą i hasłem. Jest on jednak początkowo użytkownikiem bez żadnych przywilejów, dlatego musi czekać na nadanie ich przez zarządzającego. Po weryfikacji ma dostęp do wyświetlania rekordów wszystkich tabel, ale tylko do niektórych uzyskuje prawa modyfikacji.



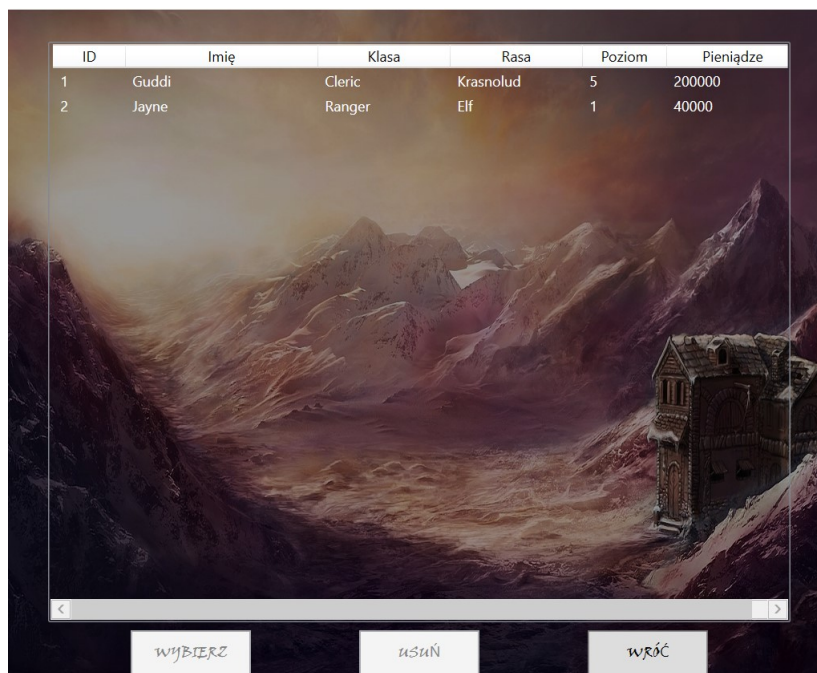
Okna logowania i rejestracji.

Po pomyślnej weryfikacji użytkownik ma dostęp do interfejsu graficznego, w którym znajdzie następujące opcje: lista postaci, widok wybranej, dodanie nowej postaci, czy edytowanie i usuwanie już istniejących.

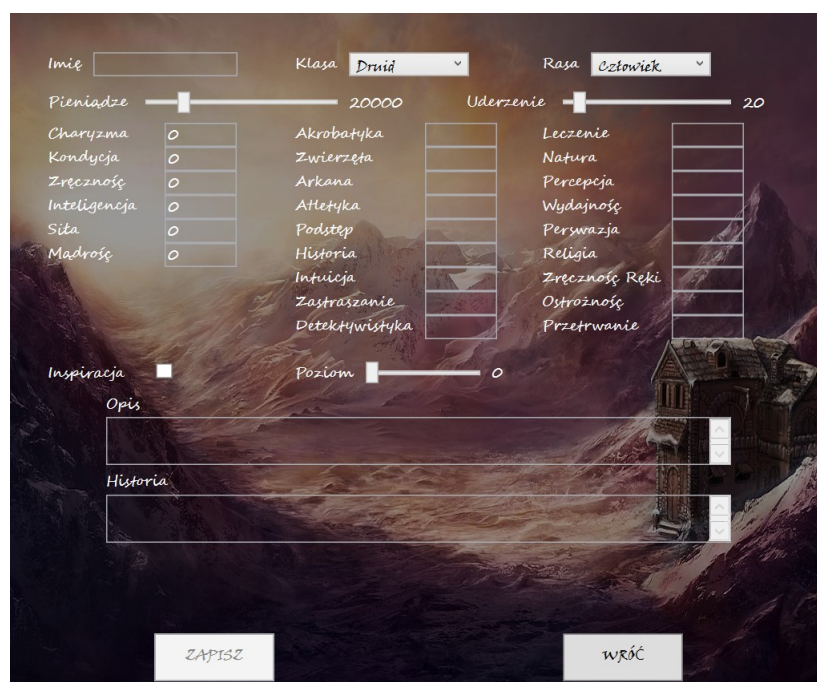


Widok głównego menu, kliknięcie myszką odpowiedniego przycisku przenosi do danej funkcjonalności.





Widok listy postaci, kliknięcie myszką odpowiedniego wiersza oraz przycisku na dole okna pozwala na wykonanie odpowiedniej funkcji - wyświetlenie widoku wybranej postaci lub jej usunięcie.



W tym oknie użytkownik może przypisać wartości odpowiednim atrybutom swojej postaci i zapisać ją do bazy danych.

## 5 Podsumowanie

### 5.1 Zrealizowane pomysły

Udało się nam połączyć aplikację z bazą danych, zostało poprawnie zaimplementowanie dodawanie, wczytywanie i usuwanie postaci wraz z jej ekwipunkiem. Użytkownik ma dostęp do listy wszystkich postaci oraz widoku jednej konkretnej.

### 5.2 Problemy

Największe problemy sprawiły nam rzeczy związane z widokiem aplikacji i połączenie go w odpowiedni sposób z warstwą modelu widoku. Szczególnie ten problem objawił się przy modelu widoku karty postaci której funkcjonalność w wyniku problemów została okrojona to kompletnego minimum.


































### 5.3 Plany na przyszłość

Przed nami stoją następujące możliwości:

1. Rozwijanie bazy pod kątem ilości elementów w niej zawartych,
2. Wprowadzenie nowych encji
3. Rozbudowanie większej i bardziej wyspecjalizowanej bazy użytkowników oraz podniesienie standardów bezpieczeństwa
4. Ewentualna optymalizacja aplikacji
5. Umieszczenie bazy na serwerze z przestrzenią dyskową tak aby można było korzystać optymalnie ze ścieżek do obrazów.

## 6 Wykorzystanie systemu kontroli wersji

Do napisania projektu używaliśmy wspólnego repozytorium git.

CharacterSheet View model part_1, Dodanie klasy Money  Mordtimer committed 2 days ago	 3ebb31a	
Dodanie możliwości logowania i rejestracji konta  PiotrHadam committed 2 days ago	 3e4806a	
Przed złączeniem  PiotrHadam committed 2 days ago	 77d1def	
Dodany widok dla karty postaci i przygotowane własności do obsłużenia...  Mordtimer committed 2 days ago	 0e529b0	
Commits on Jul 7, 2020		
Dodanie możliwości edytowania postaci  PiotrHadam committed 3 days ago	 27ae927	
Dodana obsługa listy postaci  PiotrHadam committed 3 days ago	 73287cd	
Before merging  PiotrHadam committed 3 days ago	 6ac2269	
Dodanie buildera dla klasy Character i konstruktora opartego o builde...  Mordtimer committed 3 days ago	 718451d	
Removing Conflicts  Mordtimer committed 3 days ago	 e667d42	
commit before merge  Mordtimer committed 3 days ago	 ac4ef6a	
Stworzenie widoku modelu i połączenie z nim widoku dodania postaci  PiotrHadam committed 3 days ago	 7ad30ac	

Przykładowe commity do repozytorium.