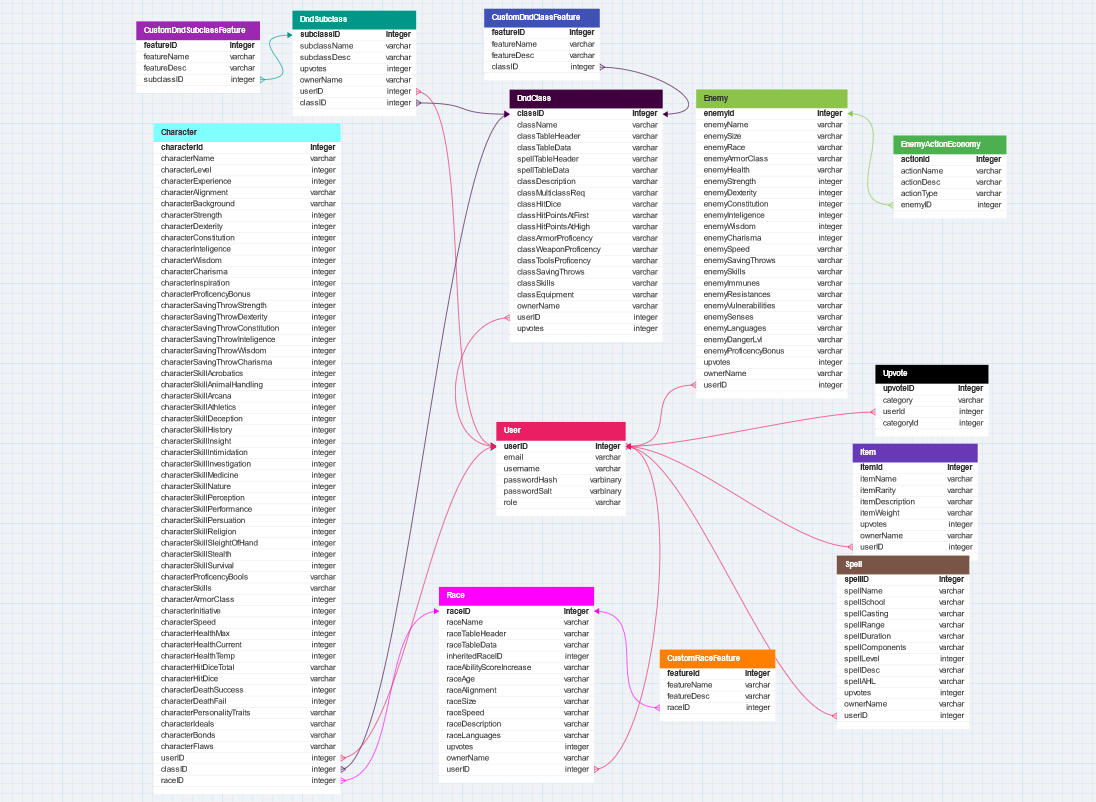
# Funkcjonalność aplikacji webowej zajmującej się logiką i integracją funkcji po stronie serwera (Backend)

## Budowa bazy danych



Dbdesigner.net

Baza danych zbudowana jest z:

1.User – Użytkownik

userID - identyfikator użytkownika

username – nazwa użytkownika

passwordHash oraz passwordSalt – sposób zakodowania hasła, aby nie było widoczne w bazie danych

role – aktualna rola użytkownika (user lub admin)

2.DndClass – klasa (profesja) postaci

classID - identyfikator klasy

className – nazwa klasy

classTableHeader oraz classTableData – sposób zapisu tabel klasy w formie znaków

spellTableHeader oraz spellTableData – sposób zapisu tabel zaklęć w formie znaków

classDescription – opis klasy

classMulticlassReq – wymagania aby postać była wieloklasowa

classHitDice – zapis kostek wymaganych aby ustalić ilość punktów życia oraz ilość odzyskiwanych podczas odpoczynków

classHitPointsAtFirst – minimalna ilość punktów życia do zdobycia

classHitPointAtHigh – maksymalna ilość punktów życia do zdobycia

classArmorProficency – biegłość klasy do danych rodzajów pancerza

classWeaponProficency – biegłość klasy do danych rodzajów broni

classSavingThrows – rzuty kostką za pomocą których stwierdzane jest czy postać obroniła się przed danym czynem

classSkills – umiejętności klasy

classEquipment – wyposażenie klasy

ownerName – nazwa twórcy (użytkownika)

userID – identyfikator twórcy (użytkownika)

upvotes – ilość polubieni klasy

3.CustomDndClassFeature – dodatkowe funkcje dla klas stworzone przez twórcę

featureID – identyfikator funkcji

featureName – nazwa funkcji

featureDesc – opis funkcji

classID – identyfikator klasy dla funkcji

4.DndSubclass – podklasa klasy

subclassID – identyfikator podklasy

subclassName – nazwa podklasy

subclassDesc – opis podklasy

upvotes – ilość polubieni podklasy

ownerName – nazwa twórcy (użytkownika)

userID – identyfikator twórcy (użytkownika)

classID – identyfikator klasy dla podklasy

5.CustomDndSubclassFeature – dodatkowe funkcje dla podklas stworzonych przez twórcę

featureID – identyfikator funkcji

featureName – nazwa funkcji

featureDesc – opis funkcji

subclassID – identyfikator podklasy dla funkcji

6.Enemy – postać przeciwnika którego DM może wykorzystać do walki z graczami

enemyID – identyfikator przeciwnika

enemyName – nazwa przeciwnika

enemySize – wzrost przeciwnika

enemyRace – rasa przeciwnika

enemyArmorClass – wartość pancerza przeciwnika którą gracze muszą przebić kostką aby go zaatakować

enemyHealth – ilość punktów życia

enemyStrength – wartość siły przeciwnika

enemyDexterity – wartość zręczności przeciwnika

enemyConstitution – wartość kondycji przeciwnika

enemyInteligence – wartość inteligencji przeciwnika

enemyWisdom – wartość mądrości przeciwnika

enemyCharisma – wartość charyzmy przeciwnika

enemySpeed – prędkość przeciwnika

enemySavingThrows - rzuty kostką za pomocą których stwierdzane jest czy przeciwnik obronił się przed danym czynem

enemySkills – umiejętności przeciwnika

enemyImmunes – niewrażliwości przeciwnika

enemyResistances – odporności przeciwnika

enemyVulnerabilities – wrażliwości przeciwnika

enemySenses – zmysły przeciwnika

enemyLanguages – znane języki przeciwnika

enemyDangerLvl – teoretyczna trudność pokonania przeciwnika

enemyProficenceBonus – biegłości przeciwnika

upvotes – ilość polubień przeciwnika

ownerName – nazwa twórcy (użytkownika)

userID – identyfikator twórcy (użytkownika)

7.EnemyActionEconomy – akcje jakich może się podjąć przeciwnik

actionID – identyfikator akcji

actionName – nazwa akcji

actionDesc – opis akcji

actionType – rodzaj akcji

enemyId – identyfikator przeciwnika dla akcji

8.Upvote – polubienia wszystkich kategorii

upvoteID – identyfikator polubienia

category – kategoria polubienia

userID – identyfikator użytkownika

categoryID – identyfikator danej kategorii

9.Item – przedmioty dla postaci lub występujące w świecie

itemID – identyfikator przedmiotu

itemName – nazwa przedmiotu

itemRarity – rzadkość przedmiotu

itemDescription – opis przedmiotu

itemWeight – waga przedmiotu

upvotes – ilośc polubień przedmiotu

ownerName – nazwa twórcy (użytkownika)

userID – identyfikator twórcy (użytkownika)

10.Spell – zaklęcia postaci

spellID – identyfikator zaklęcia

spellName – nazwa zaklęcia

spellSchool – kategoria zaklęcia

spellCasting – długość procesu rzucenia zaklęcia

spellRange – odległość na jaką zaklęcie może zostać rzucone

spellDuration – długość trwania zaklęcia

spellComponents – składniki potrzebne do rzucenia zaklęcia

spellLevel – stopień zaklęcia

spellDesc – opis zaklęcia

spellAHL – dodatkowe efekty jeśli zaklęcie zostanie rzucone w wyższym stopniu niż podstawowe

upvotes – ilość polubieni zaklęcia

ownerName – nazwa twórcy (użytkownika)

userID – identyfikator twórcy (użytkownika)

11.Race – rasa postaci

raceID – identyfikator rasy

raceName – nazwa rasy

raceTableHeader oraz raceTableData - sposób zapisu tabel rasy w formie znaków

inheritedRaceID – identyfikator rasy na której podstawie została stworzona aktualna

raceAbilityScoreIncrease – dodatkowe punkty przy wyborze tej rasy dla postaci w formie statystyk

raceAge – wiek rasy

raceAlignment – etyczna oraz moralna przynależność rasy

raceSize – wzrost rasy

raceSpeed – prędkość rasy

raceDescription – opis rasy

upvotes – ilość polubień rasy

ownerName – nazwa twórcy (użytkownika)

userID – identyfikator twórcy (użytkownika)

12.CustomRaceFeature – dodatkowe funkcje dla ras stworzonych przez twórcę

featureID – identyfikator funkcji

featureName – nazwa funkcji

featureDesc – opis funkcji

raceID – identyfikator rasy dla funkcji

13.Character – karta postaci

characterID – identyfikator postaci

characterName – nazwa postaci

characterLevel – poziom postaci

characterExperience – ilość doświadczenia postaci

characterAlignment - etyczna oraz moralna przynależność postaci

characterBackground – przeszłość postaci

characterStrength – wartość siły postaci

characterDexterity – wartość zręczności postaci

characterConstitution – wartość kondycji postaci

characterInteligence – wartość inteligencji postaci

characterWisdom – wartość mądrości postaci

characterCharisma – wartość charyzmy postaci

characterInspiration – dodatkowe punkty w rzutach kostką podarowane przez GM

characterProficencyBonus – liczba punktów dodawana przy rzutach kostką na dane umiejętności

characterSavingThrowStrength – dodatkowe punkty przy rzutach kostką za pomocą siły na obronę przed czynem

characterSavingThrowDexterity – dodatkowe punkty przy rzutach kostką za pomocą zręczności na obronę przed czynem

characterSavingThrowConstitution – dodatkowe punkty przy rzutach kostką za pomocą kondycji na obronę przed czynem

characterSavingThrowInteligence – dodatkowe punkty przy rzutach kostką za pomocą inteligencji na obronę przed czynem

characterSavingThrowWisdom – dodatkowe punkty przy rzutach kostką za pomocą mądrości na obronę przed czynem

characterSavingThrowCharisma – dodatkowe punkty przy rzutach kostką za pomocą charyzmy na obronę przed czynem

characterSkillAcrobatics – wartość umiejętności akrobatyki postaci

characterSkillAnimalHandling – wartość umiejętności opieki nad zwierzętami postaci

characterSkillArcana – wartość umiejętności wiedzy tajemnej postaci

characterSkillAthletics – wartość umiejętności atletyki postaci

characterSkillDeception – wartość umiejętności oszustwa postaci

characterSkillHistory – wartość umiejętności znajomości historii postaci

characterSkillInsight – wartość umiejętności intuicji postaci

characterSkillInitimidation – wartość umiejętności zastraszania postaci

characterSkillInvestigation – wartość umiejętności śledztwa postaci

characterSkillMedicine – wartość umiejętności medycyny postaci

characterSkillNature – wartość umiejętności znajomości natury postaci

characterSkillPerception – wartość umiejętności percepcji postaci

characterSkillPerformance – wartość umiejętności występów postaci

characterSkillPersuation – wartość umiejętności perswazji postaci

characterSkillReligion – wartość umiejętności znajomości religii postaci

characterSkillSleightOfHand – wartość umiejętności zwinnych dłoni postaci

characterSkillStealth – wartość umiejętności ukrywania się postaci

characterSkillSurvival – wartość umiejętności sztuki przetrwania postaci

characterProficencyBools - sposób zapisu tabel biegłości postaci w formie znaków

characterSkills – dodatkowe umiejętności postaci

characterArmorClass - wartość pancerza postaci którą przeciwnicy muszą przebić kostką aby go zaatakować

characterInitiative – dodatkowe punkty przy rzucie kostką na kolejność tur w walce

characterSpeed – prędkość postaci

characterHealthMax – maksymalna ilość punktów życia postaci

characterHealthCurrent – aktualna ilość punktów życia postaci

characterHealthTemp – dodatkowa ilość punktów życia postaci

characterHitDiceTotal - całkowita ilość kostek wymaganych aby ustalić ilość punktów życia oraz ilość odzyskiwanych podczas odpoczynków

characterHitDice – aktualna ilość kostek wymaganych aby ustalić ilość punktów życia oraz ilość odzyskiwanych podczas odpoczynków

characterDeathSuccess – ilość udanych rzutów na przetrwanie postaci w fazie nieświadomości przed śmiercią

characterDeathFail – ilość nieudanych rzutów na przetrwanie postaci w fazie nieświadomości przed śmiercią

characterPersonalityTraits – cechy charakteru postaci

characterIdeals – ideały postaci

characterBonds –więzi postaci

characterFlaws – wady postaci

userID – identyfikator twórcy (użytkownika)

classID – identyfikator aktualnej klasy postaci

raceID – identyfikator aktualnej rasy postaci

Relacje w bazie danych:

1. User i DndClass – relacja jeden do wielu. Jeden użytkownik może mieć wiele klas.

2.DndClass i CustomDndClassFeatures – relacja jeden do wielu. Jedna klasa może mieć wiele funkcji.

3.User i DndSubclass – relacja jeden do wielu. Jeden użytkownik może mieć wiele podklas.

4.DndSubclass i CustomDndSubclassFeature – relacja jeden do wielu. Jedna podklasa może mieć wiele funkcji

5.DndClass i DndSubclass – relacja jeden do wielu. Jedna klasa może mieć wiele podklas

6.User i Enemy – relacja jeden do wielu. Jeden użytkownik może mieć wiele przeciwników.

7.Enemy i EnemyActionEconomy – relacja jeden do wielu. Jeden przeciwnik może mieć wiele akcji

8.User i Upvote – relacja jeden do wielu. Jeden użytkownik może mieć wiele polubień.

9.User i Item – relacja jeden do wielu. Jeden użytkownik może mieć wiele przedmiotów.

10.User i Spell – relacja jeden do wielu. Jeden użytkownik może mieć wiele zaklęć.

11. User i Race – relacja jeden do wielu. Jeden użytkownik może mieć wiele ras.

12.Race i CustomRaceFeature – relacja jeden do wielu. Jedna rasa może mieć wiele funkcji.

13.User i Character – relacja jeden do wielu. Jeden użytkownik może mieć wiele postaci.

14.DndClass i Character – relacja jeden do wielu. Jedna klasa może należeć do wielu postaci.

15.Race i Character – relacja jeden do wielu. Jedna rasa może należeć o wielu postaci.

## Budowa Backendu

Link do aktywnego backendu uruchomionego na serwerach Microsoft to: <https://kumpleismokibbkservice.azurewebsites.net>

Bezpieczeństwo:

Po zalogowaniu do serwisu, otrzymuje się swój własny token do identyfikacji aktualnego użytkownika. Token ten zawiera identyfikator serwera oraz stwierdzenia na podstawie których następuje identyfikacja użytkownika. Stwierdzeniami tymi są nazwa użytkownika oraz jego rola. Na podstawie tych dwóch wartości backend określa z jakim użytkownikiem ma do czynienia.

Podczas rejestracji, hasło użytkownika jest kodowane w bazie danych, za pomocą kryptografii SHA256.



Generuje ona losowy klucz bajtów (Salt) przy każdej rejestracji i na jego podstawie oblicza z hasła wartość ciągu znaków w bajtach(Hash). Dzięki temu każde hasło zapisane w bazie danych jest inaczej zaszyfrowane, co sprawia że w przypadku tych samych haseł nie ma możliwości tego zauważenia bez ich dekodowania.

Na potrzeby skrócenia odnośników aktywny backend będzie nazywany „host”.

Spis żądań HTTPS:

1.User

Metoda GET **host/api/User** – Zwraca listę użytkowników. Metoda ta jest zastrzeżona przed zwykłymi użytkownikami za pomocą wymagania tokenu użytkownika posiadającego uprawnienia „admin”.

Metoda GET **host/api/User/user** – Zwraca dane danego użytkownika na podstawie otrzymanego tokenu. Celem tej metody jest przekazanie danych frontendowi nt. aktualnie zalogowanego użytkownika.

Metoda GET **host/api/User/{id}** – Pobiera od użytkownika zmienną id i na jej podstawie zwraca dane użytkownika o podanym identyfikatorze. Ta metoda nie ma restrykcji, gdyż nie wysyła danych wrażliwych typu hasło lub email.

Metoda POST **host/api/User/register** – Rejestruje nowego użytkownika najpierw sprawdzając czy podany email oraz nazwa użytkownika nie są już zajęte.

Metoda POST **host/api/User/login** – Weryfikuje email oraz hasło użytkownika w celu odesłania jego tokenu.

Metoda DELETE **host/api/User/{id}** – Usuwa użytkownika z bazy danych na podstawie podanego id. Dostęp do niej mają użytkownicy z rolą „admin”.

Metoda GET **host/api/User/edit/username** – Zmienia nazwę użytkownika w sytuacji gdy nowa nazwa nie jest już zajęta.

Metoda POST **host/api/User/edit/password** – Edytuje hasło użytkownika na nowe jeśli udało mu się zweryfikować stare hasło.

2.DndClass

Metoda GET **host/api/DndClass** – Zwraca listę wszystkich klas. Jest ona ogólnodostępna, ponieważ nie zawiera żadnych informacji wrażliwych.

Metoda GET **host/api/DndClass/id/{id}** – Zwraca daną klasę na podstawie podanego id.

Metoda GET **host/api/DndClass/name/{classname}** – Zwraca listę klas posiadających podaną nazwę classname.

Metoda GET **host/api/DndClass/owner/{ownerid}** – Zwraca listę klas utworzonych przez użytkownika z identyfikatorem podanym przez ownerid

Metoda GET **host/api/DndClass/subclass/{classid}** – Zwraca listę podklas które są utworzone na podstawie klasy o identyfikatorze podanym w classid.

Metoda GET **host/api/DndClass/upvote/{classid}** – Dodaje lub usuwa z ulubionych użytkownika klasę z podanym classid. Następnie modyfikuje aktualną ilość całkowitych polubień danej klasy.

Metoda GET **host/api/DndClass/checkifupvote/{classid}** – Sprawdza czy aktualny użytkownik ma już dodane do ulubionych daną klasę. Metoda ta ma na celu pomóc frontendowi określić stan przycisku dodawania lub usuwania z ulubionych podczas ładowania strony.

Metoda GET **host/api/DndClass/upvoted/** - Zwraca listę dodanych do ulubionych klas aktualnego użytkownika.

Metoda POST **host/api/DndClass** – Dodaje nową klasę utworzoną przez użytkownika.

Metoda PUT **host/api/DndClass** – Modyfikuje daną klasę należącą do użytkownika.

Metoda DELETE **host/api/DndClass** – Usuwa podaną klasę utworzoną przez użytkownika z bazy danych.

3.CustomDndClassFeature

Metoda GET **host/api/CustomDndClassFeature** – Zwraca listę wszystkich funkcji klas

Metoda GET **host/api/CustomDndClassFeature/{classid}** – Zwraca listę funkcji klas dla podanego id klasy.

Metoda POST **host/api/CustomDndClassFeature** – Dodaje nową funkcję do klasy utworzoną przez użytkownika.

Metoda PUT **host/api/CustomDndClassFeature** – Modyfikuje daną funkcję należącą do użytkownika.

Metoda DELETE **host/api/CustomDndClassFeature** – Usuwa aktualną funkcję utworzoną przez użytkownika w bazie danych.

4.DndSubclass

Metoda GET **host/api/DndSubclass** – Zwraca listę wszystkich podklas.

Metoda GET **host/api/DndSubclass/id/{subclassid}** – Zwraca podklasę o podanym id.

Metoda GET **host/api/DndSubclass/owner/{ownerid}** – Zwraca listę podklas utworzonych przez użytkownika z identyfikatorem podanym przez ownerid

Metoda GET **host/api/DndSubclass/class/{subclassid}** – Zwraca klasę od której dana podklasa dziedziczy na podstawie id.

Metoda GET **host/api/DndSubclass/upvote/{subclassid}** - Dodaje lub usuwa z ulubionych użytkownika podklasę z podanym subclassid. Następnie modyfikuje aktualną ilość całkowitych polubień danej podklasy.

Metoda GET **host/api/DndSubclass/checkifupvote/{subclassid}** – Sprawdza czy aktualny użytkownik ma już dodane do ulubionych daną podklasę. Metoda ta ma na celu pomóc frontendowi określić stan przycisku dodawania lub usuwania z ulubionych podczas ładowania strony.

Metoda GET **host/api/DndSubclass/upvoted/** - Zwraca listę dodanych do ulubionych podklas aktualnego użytkownika.

Metoda POST **host/api/DndSubclass** – Dodaje nową podklasę utworzoną przez użytkownika.

Metoda PUT **host/api/DndSubclass** – Modyfikuje daną podklasę należącą do użytkownika.

Metoda DELETE **host/api/DndSubclass** – Usuwa podaną podklasę utworzoną przez użytkownika z bazy danych.

5.CustomDndSubclassFeature

Metoda GET **host/api/CustomDndSubclassFeature** – Zwraca listę wszystkich funkcji podklas.

Metoda GET **host/api/CustomDndSubclassFeature/{subclassid}** - Zwraca listę funkcji podklas dla podanego id podklasy.

Metoda POST **host/api/CustomDndSubclassFeature** – Dodaje nową funkcję do podklasy utworzoną przez użytkownika.

Metoda PUT **host/api/CustomDndSubclassFeature** – Modyfikuje daną funkcję należącą do użytkownika.

Metoda DELETE **host/api/CustomDndSubclassFeature** – Usuwa aktualną funkcję utworzoną przez użytkownika w bazie danych.

6. Enemy

Metoda GET **host/api/Enemy** – Zwraca listę obiektów przeciwników.

Metoda GET **host/api/Enemy/owner/{ownerid}** – Zwraca listę przeciwników utworzonych przez użytkownika o podanym ownerid.

Metoda GET **host/api/Enemy/id/{enemyid}** – Zwraca przeciwnika o podanym id.

Metoda GET **host/api/Enemy/upvote/{enemyid}** - Dodaje lub usuwa z ulubionych użytkownika przeciwnika z podanym enemyid. Następnie modyfikuje aktualną ilość całkowitych polubień danego przeciwnika.

Metoda GET **host/api/Enemy/checkifupvote/{enemyid}** - Sprawdza czy aktualny użytkownik ma już dodane do ulubionych danego przeciwnika. Metoda ta ma na celu pomóc frontendowi określić stan przycisku dodawania lub usuwania z ulubionych podczas ładowania strony.

Metoda GET **host/api/Enemy/upvoted/** - Zwraca listę dodanych do ulubionych przeciwników aktualnego użytkownika.

Metoda POST **host/api/Enemy** – Dodaje nowego przeciwnika utworzonego przez użytkownika.

Metoda PUT **host/api/Enemy** – Modyfikuje danego przeciwnika należącego do użytkownika.

Metoda DELETE **host/api/Enemy** – Usuwa podanego przeciwnika utworzonego przez użytkownika z bazy danych.

7.EnemyActionEconomy

Metoda GET **host/api/EnemyActionEconomy** – Zwraca listę akcji przeciwników.

Metoda GET **host/api/EnemyActionEconomy/enemy/{enemyid}** – Zwraca listę akcji przeciwnika o podanym enemyid.

Metoda POST **host/api/EnemyActionEconomy** – Dodaje nową akcję do przeciwnika utworzonego przez użytkownika.

Metoda PUT **host/api/EnemyActionEconomy** – Modyfikuje daną akcję należącą do użytkownika.

Metoda DELETE **host/api/EnemyActionEconomy** – Usuwa aktualną akcję utworzoną przez użytkownika w bazie danych.

8.Item

Metoda GET **host/api/Item** – Zwraca listę wszystkich przedmiotów.

Metoda GET **host/api/Item/id/{itemid**} – Zwraca przedmiot o podanym itemid.

Metoda GET **host/api/Item/owner/ownerid** – Zwraca listę przedmiotów utworzonych przez użytkownika o podanym ownerid.

Metoda GET **host/api/Item/upvote/{itemid}** - Dodaje lub usuwa z ulubionych użytkownika przedmiot z podanym itemid. Następnie modyfikuje aktualną ilość całkowitych polubień danego przedmiotu.

Metoda GET **host/api/Item/checkifupvote/{itemid}** - Sprawdza czy aktualny użytkownik ma już dodane do ulubionych dany przedmiot. Metoda ta ma na celu pomóc frontendowi określić stan przycisku dodawania lub usuwania z ulubionych podczas ładowania strony.

Metoda GET **host/api/Item/upvoted/** - Zwraca listę dodanych do ulubionych przedmiotów aktualnego użytkownika.

Metoda POST **host/api/Item** – Dodaje przedmiot utworzony przez użytkownika do bazy danych.

Metoda PUT **host/api/Item** – Modyfikuje dany przedmiot należący do użytkownika.

Metoda DELETE **host/api/Item** – Usuwa podany przedmiot utworzony przez użytkownika z bazy danych.

9. Spell

Metoda GET **host/api/Spell** – Zwraca listę wszystkich zaklęć.

Metoda GET **host/api/Spell/id/{spellid}** – Zwraca zaklęcie o podanym spellid.

Metoda GET **host/api/Spell/name/{spellname}** – Zwraca listę zaklęć o podanej nazwie spellname.

Metoda GET **host/api/Spell/lvl/{spelllevel}** – Zwraca listę zaklęć które posiadają poziom spelllevel.

Metoda GET **host/api/Spell/owner/spell/{spellid}** – Zwraca użytkownika który utworzył zaklęcie o identyfikatorze spellid.

Metoda GET **host/api/Spell/owner/{ownerid}** – Zwraca listę zaklęć utworzonych przez użytkownika o identyfikatorze ownerid.

Metoda GET **host/api/Spell/upvote/{spellid}** - Dodaje lub usuwa z ulubionych użytkownika zaklęcie z podanym spellid. Następnie modyfikuje aktualną ilość całkowitych polubień danego zaklęcia.

Metoda GET **host/api/Spell/checkifupvote/{spellid}** - Sprawdza czy aktualny użytkownik ma już dodane do ulubionych dane zaklęcie. Metoda ta ma na celu pomóc frontendowi określić stan przycisku dodawania lub usuwania z ulubionych podczas ładowania strony.

Metoda GET **host/api/Spell/upvoted/** - Zwraca listę dodanych do ulubionych zaklęć aktualnego użytkownika.

Metoda POST **host/api/Spell** – Dodaje nowe zaklęcie utworzone przez użytkownika.

Metoda PUT **host/api/Spell** – Modyfikuje dane zaklęcie należące do użytkownika.

Metoda DELETE **host/api/Spell** – Usuwa podane zaklęcie utworzone przez użytkownika z bazy danych.

10. Race

Metoda GET **host/api/Race** – Zwraca listę wszystkich ras.

Metoda GET **host/api/Race/id/{raceid}** – Zwraca rasę o identyfikatorze raceid.

Metoda GET **host/api/Race/name/{racename}** – Zwraca listę ras które mają nazwę racename.

Metoda GET **host/api/Race/owner/{ownerid}** – Zwraca listę ras utworzonych przez użytkownika o identyfikatorze ownerid.

Metoda GET **host/api/Race/upvote/{raceid}** - Dodaje lub usuwa z ulubionych użytkownika rasę z podanym raceid. Następnie modyfikuje aktualną ilość całkowitych polubień danej rasy.

Metoda GET **host/api/Race/checkifupvote/{raceid}** - Sprawdza czy aktualny użytkownik ma już dodane do ulubionych daną rasę. Metoda ta ma na celu pomóc frontendowi określić stan przycisku dodawania lub usuwania z ulubionych podczas ładowania strony.

Metoda GET **host/api/Race/upvoted/** - Zwraca listę dodanych do ulubionych ras aktualnego użytkownika.

Metoda POST **host/api/Race** - Dodaje nową rasę utworzoną przez użytkownika.

Metoda PUT **host/api/Race**– Modyfikuje daną rasę należącą do użytkownika.

Metoda DELETE **host/api/Race** – Usuwa podaną rasę utworzoną przez użytkownika z bazy danych.

11. CustomRaceFeature

Metoda GET **host/api/CustomRaceFeature** – Zwraca listę funkcji ras.

Metoda GET **host/api/CustomRaceFeature/{raceid}** – Zwraca listę funkcji ras dla rasy o identyfikatorze raceid.

Metoda POST **host/api/CustomRaceFeature** – Dodaje nową funkcję do rasy utworzoną przez użytkownika.

Metoda PUT **host/api/CustomRaceFeature** – Modyfikuje daną funkcję należącą do użytkownika.

Metoda DELETE **host/api/CustomRaceFeature** – Usuwa aktualną funkcję utworzoną przez użytkownika w bazie danych.

12.Character

Metoda GET **host/api/Character** – Zwraca listę wszystkich postaci.

Metoda GET **host/api/Character/id/{id}** – Zwraca postać o identyfikatorze id.

Metoda GET **host/api/Character/owner/{ownerid}** – Zwraca listę postaci utworzonych przez użytkownika o identyfikatorze ownerid.

Metoda POST **host/api/Character** - Dodaje nową postać utworzoną przez użytkownika.

Metoda PUT **host/api/Character**– Modyfikuje daną postać należącą do użytkownika.

Metoda DELETE **host/api/Race** – Usuwa podaną postać utworzoną przez użytkownika.

Technologie

Wzorzec projektowy

Rest API to zbiór reguł określający jak powinno budować się interfejs programistyczny API, który umożliwia przesyłanie danych pomiędzy aplikacją a serwerem webowym. Aby backend był na standardzie REST musi mieć:

-Niezależny interfejs od urządzenia tzn. przesyłać dane do urządzeń mobilnych lub aplikacji webowych tymi samymi metodami.

-Podział między aplikacją klienta oraz serwera.

-Całość informacji potrzebnych do ukończenia żądań w każdym zapytaniu.

Język programowania C#

To język obiektowy stworzony przez firmę Microsoft używający środowisk .NET. Pochodzi z języka C oraz jest bardzo podobny do innych języków takich jak: Java, Object Pascal czy C++. Używany jest w aplikacjach konsolowych, internetowych itd.

Entity Framework

To biblioteka używana do budowania aplikacji które są oparte na bazach danych. Pozwala ona na zamianę tabel oraz danych na obiekty klas w programie.

AspNetCore

To framework umożliwiający wytwarzanie internetowych aplikacji połączonych z obsługą chmury. Charakteryzuje się wieloplatformowością, open source oraz architekturą zaprojektowaną na możliwość przeprowadzania testów jednostkowych.

AutoMapper

To biblioteka pozwalająca na automatyczne kopiowanie oraz przemianę jednego obiektu do drugiego. Pozwala również na ustalenie w jaki sposób zmieniać dane rodzaje zmiennych.

Visual Studio 2022

To środowisko IDE stworzone przez Microsoft pozwalające na pisanie, edycje, debugowanie, kompilacje oraz publikowanie programów. Pozwala na programowanie międzyplatformowe, używanie kontroli źródła i narzędzi do uzupełniania kodu.

Azure

To platforma firmy Microsoft umożliwiająca hosting serwerów, baz danych SQL i 200 innych produktów oraz usług w chmurze. Pozwala na darmowe założenie konta dla którego część usług jest bezpłatna przez rok.

Azure SQL Database

To usługa umożliwiająca założenie bazy danych SQL oraz jej hosting. Zbudowana jest na bazie danych programu Microsoft SQL Server. Posiada wbudowaną funkcję PaaS pozwalającą na administrację bazami danych.