DBS Zadanie 2

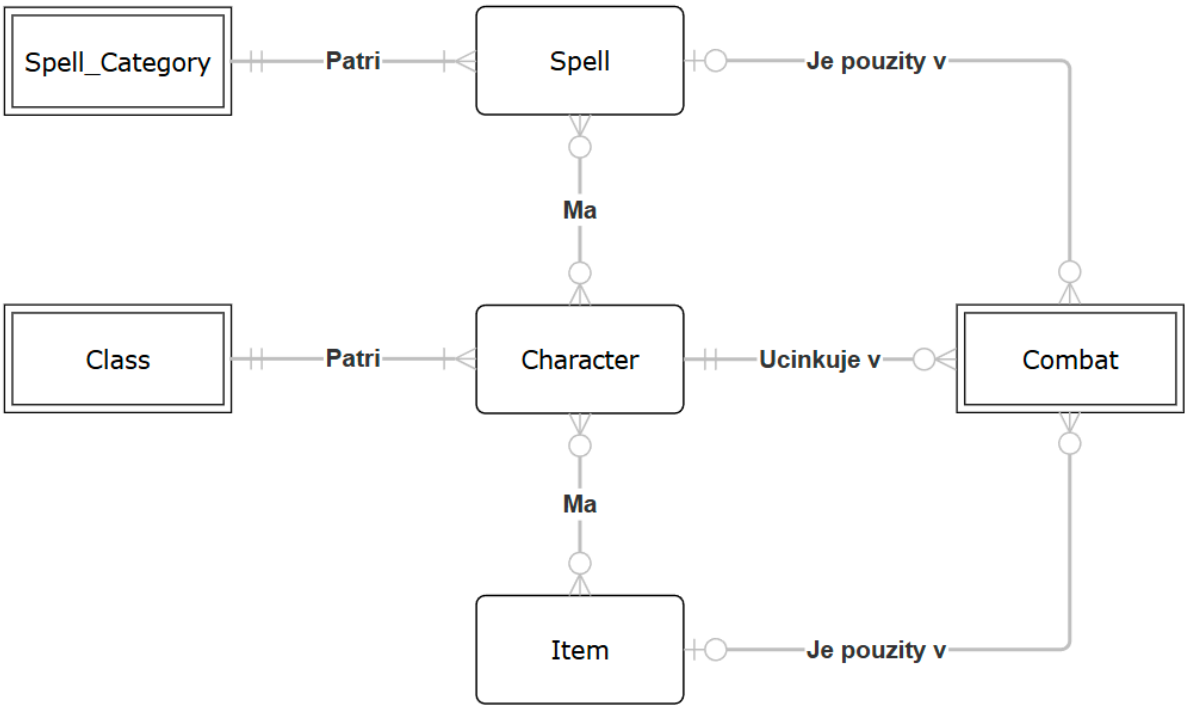
Dokumentácia

Rodion Burmistrov

V rámci Zadania číslo 2 bolo potrebné navrhnúť databázu pre RPG combat hru. Cieľom bolo vytvoriť flexibilný systém s postavami, spellami, systémom bitiek, inventármi a ekonomikou akčných bodov. Ďalej v dokumente je popísaný navrhnutý systém, jeho entity a ich vzťahy, vrátane požadovaných procesov.

1.ER diagram

Na začiatku projektu bolo potrebné vytvoriť ER diagram, ktorý bude kostrou návrhu a z ktorého budeme vychádzať pri tvorbe logického modelu s entitami a vzťahmi. Výsledkom je ER diagram, ktorý som vytvoril v aplikácii SmartDraw, pričom som použil notáciu Crow's Foot.



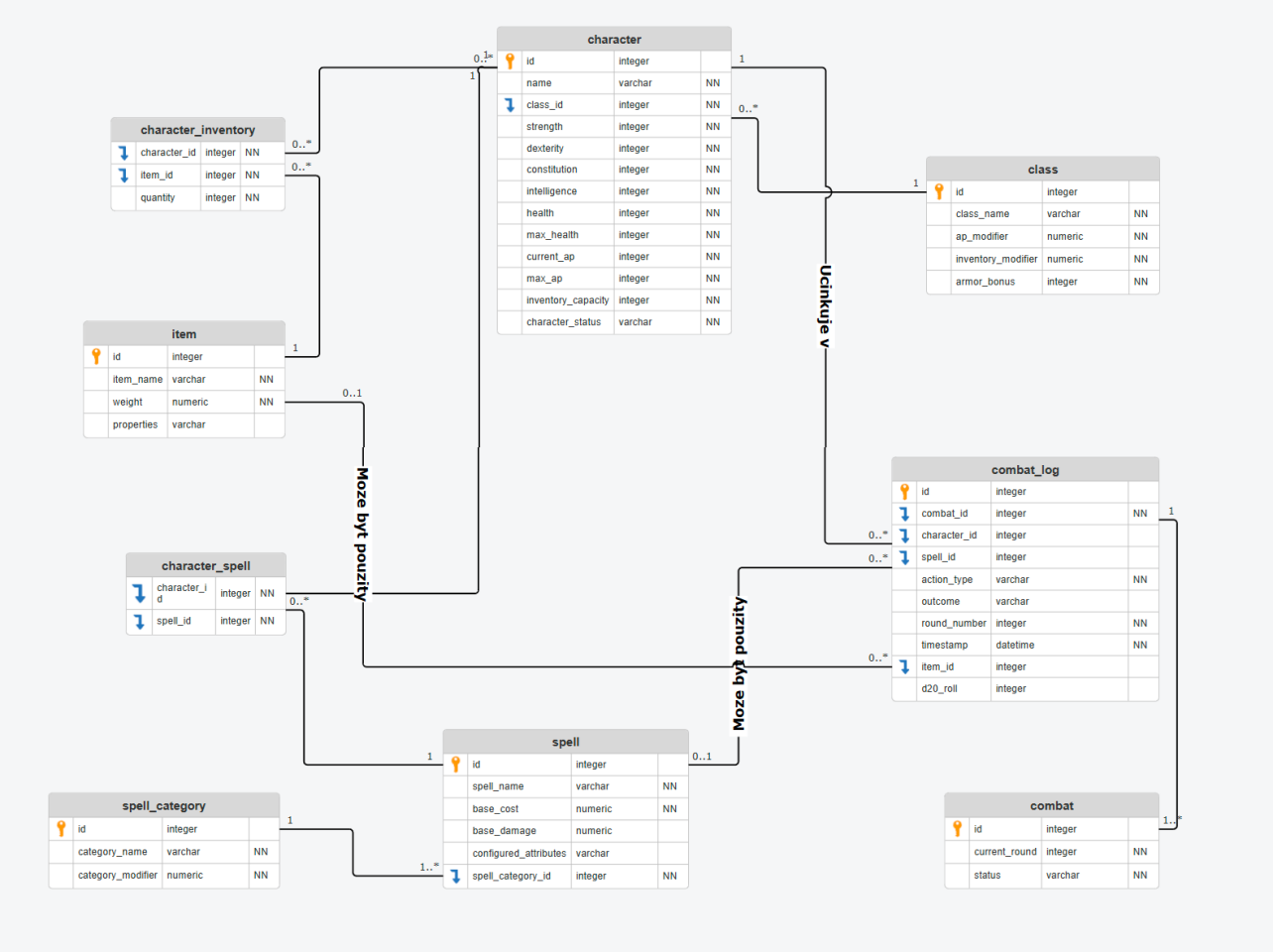
Hlavnou myšlienkou tohto diagramu teda je:

- Existuje postava (Character), ktorá patrí do určitej triedy (Class). Jednu konkrétnu triedu môže mať viacero postáv.
- Postava môže používať rôzne spelly, ktoré ale nevymyslela sama. Samozrejme, tie isté spelly môže mať viacero postáv. V budúcnosti budeme môcť pridať aj nejaký mystický spell, ktorý bude existovať, ale možno ho ani nikto nenájde.
- Spely patria do vlastných kategórií, napríklad ohnivá mágia alebo vodná mágia, pričom každá kategória má svoje charakteristické vlastnosti. Pamätáme však, že napríklad ohnivá mágia môže obsahovať viacero spellov.
- Postava môže mať aj nejaké predmety (Items), ktoré slúžia ako jej pomôcky – napríklad zbrane, liečivé elixíry a podobne. Predmet môže mať viacero majiteľov (napríklad elixír nebude patriť iba jednému hráčovi v celej hre), alebo môže ísť o legendárny mec, ktorý zatiaľ nikto nemá. Ale hráč ani nemusí mať predmety, aby prežil.
- Postava sa môže zúčastňovať bitiek. Každý krok postavy a jej oponenta bude sledovaný v boji (Combat).

- Počas bitky môže hráč používať a zároveň zbierať predmety. Je však dôležité, že počas jedného kola môže postava vybrať iba jeden predmet, no počas celej bitky môže ten istý predmet použiť viackrát.
- Rovnaký princíp platí aj pre spely. Hráč môže v rámci jedného kola použiť iba jeden spell, no počas bitky ho môže používať, kým má dostatok energie.

2.Relačný diagram

Relačný diagram som vytvoril v aplikácii SmartDraw, ktorá umožnila jednoduchšiu tvorbu diagramu pomocou predpripravených šablón. Pri tvorbe som sa snažil dodržiavať UML notáciu.



CLASS

Táto tabuľka ukladá každú triedu postáv dostupnú v hre. Obsahuje unikátny identifikátor, názov triedy (class_name, napr. „Bojovník“ alebo „Mág“), modifikátor akčných bodov (ap_modifier), ktorý ovplyvňuje počet akčných bodov dostupných pre akcie, modifikátor inventára ovplyvňujúci kapacitu prenášania predmetov (inventory_modifier) a bonus k brneniu (armor_bonus) používaný pri defenzívnych výpočtoch.

CHARACTER

Tabuľka reprezentuje postavy v hre. Každý záznam obsahuje identifikátor (*id*), meno postavy (*name*) a odkaz na tabuľku tried (*class_id*) na určenie jej triedy (ďalej sa bude používať skratka Foreign Key – FK a Primary Key - PK). Ďalšie atribúty zahŕňajú fyzické a mentálne štatistiky ako sila (*strength*), obratnosť (*dexterity*), odolnosť (*constitution*), inteligencia (*intelligence*), aktuálne

a maximálne zdravie (*health, max_health*), aktuálne a maximálne akčné body (*current_ap*, *max_ap*) a celkovú kapacitu inventára (*inventory_capacity*). Tak isto postava ma aj status (*character_status*), ktorý hovorí, zem ci je v hre, alebo je mimo hry, alebo je mrtva.

SPELL_CATEGORY

Táto tabuľka slúži na zoskupenie jednotlivých spellov do kategórií. Každá kategória má PK, názov (*category_name* napr. „Ohňová mágia“ alebo „Liečenie“) a kategóriový modifikátor (*category_modifier*), ktorý môže meniť ceny spellov v danej kategórii.

SPELL

Táto tabuľka obsahuje detaily pre každý spell v hre. Každý záznam spellu má PK, názov spellu (spell_name), základnu cenu spellu za AP (*base_cost*), základnú hodnotu utoku (*base_damage*) a textové pole (*configured_attributes*) s atribútmi ovplyvňujúcimi výkon spellu (napriklad Intelligence, dexterity). Každý spell tiež odkazuje na svoju kategóriu cez *spell_category_id*, čím je spojený so skupinou globálnych modifikátorov.

ITEM

Tabuľka obsahuje všetky predmety alebo vybavenie dostupné v hre. Každý záznam má PK, názov predmetu (*item_name* napr. „Liečivý elixír“ alebo „Obyčajný Mec“), váhu (*weight*) predmetu, na určenie, či ho postava unesie a pole s efektom ktoré obsahuje číslo (*properties,* napr. „10“ „20“). Effekt Itemu bude záležať na jeho id. Ak je id do 10, tak je healovacie. Ak v rozsahu medzi 20-30, tak pridáva AP pointy, atd.

CHARACTER_SPELL

Táto spojovacia tabuľka reprezentuje vzťah „množina k množine“ medzi postavami a spellmi, pričom určuje, ktoré spely má postava naučené alebo môže použiť. Každý záznam pozostáva z *character_id* a *spell_id*, tvoriaciu pivotnu tabulku, ktora zaistuje flexibilnost databasy a unikatnost jednotlivych parov.

CHARACTER_INVENTORY

Eviduje, ktoré predmety majú postavy pri sebe, tak isto, ako predchádzajúca tabuľka reprezentuje vzťah „množina k množine“ medzi postavami a predmetmi. Každý záznam obsahuje *character_id* a *item_id*, tvoriacie pivotnu tabulku, a pole *quantity*, ktoré udáva počet daného predmetu v inventári postavy.

COMBAT

Tabuľka COMBAT reprezentuje individuálny bojový stret. Každý stret má unikátny *combat_id* a tabuľka ukladá aktuálne číslo kola (*current_round*) a stav boja (*status*), napr. „caka na PVP“, „aktívny“, alebo „ukončený“. Reprezentuje ako keby Lobby, ktoré hrací budú môcť použiť, aby nájsť niekoho na duel.

COMBAT_LOG

Táto tabuľka zaznamenáva každú akciu alebo udalosť počas boja. Každý záznam má unikátny PK a odkazuje na bojový stret cez *combat_id*. Obsahuje aj *character_id* na identifikáciu vykonávateľa akcie. Ak akcia zahŕňa použitie spellu, použije sa pole *spell_id*, a pre akcie s predmetmi (napr. zobratie alebo vyhodenie) sa použije *item_id*. Okrem cudzích kľúčov tabuľka ukladá *action_type* (typ akcie, napr. „útok“, „zobratie predmetu“, „použitie spelu“), *outcome*, teda popis výsledku, *round_number* (kolo, v ktorom sa akcia uskutočnila), *d20_roll*, ktorý pridáva

element náhodnosti, na ktorom bude záležať *outcome,* a automaticky zaznamenaná timestamp (čas vykonania akcie).

3.Ukážkové Procesy

Na preukázanie samotných procesov bol mnou zvolený UML Sekvenčný Diagram. Je ľahko čitateľný a veľmi dobre preukazuje funkčnosť procesov vnútri Systému.

3.1 Kompletne spracovanie Kúzla v bitke (1 a 4 scenár podlá zadania)

Kde postava sa bude snažiť použiť spell v bitke – Systém najprv zoberie údaje postavy, z tabuľky Characters ako sú *current_ap, strength, dexterity, intelligence* atd. Ďalej Systém zoberie z príslušnej postave tabuľky Class modifikátor AP (ap_modificator). Následne z tabuľky Spell údaje typu cena spellu, jeho damage a atribút, na základe ktorého ma scale (*base_cost, base_damage, configured_attributes*). Systém sa tak isto pozrie aj na spell_category tabuľku, odkiaľ vytuhne category_modifier, ktorý ovplyvni cenu a damage spellu ďalej. Ma to ten zmysel, že napríklad existuje spell „Elemental_Bomb“ ktorý bude mat aj Fire a aj Lightning magia. Nech majú rovnaký dajaké a aj cenu AP. Ale tým, že Lignhting je viac zriedkavý element – bude mat sám o sebe vyšší modifier, ktorý ovplyvni aj jeho finálnu cenu a aj jeho finálny damage. Tak isto databasa zisti – aký udaj *configured_attributes* ovplyvňujúci spell musí brat od údajov Charactera (ci je to Itelligence, Strength, alebo ešte niečo).

Keď už Systém ma tieto údaje – vypočíta cenu spellu cez vzorec:

$$EffectiveCost = spell.base\_cost * spell\_category.category\_modifier * (1 - configured\_attribute\_value / 100) * ap\_modifier$$

Tato hodnota sa ďalej porovná s aktuálnym množstvom AP postavy a scenár sa rozdeli na dve možnosti.

1. Nemá dostatok current_ap na cast spellu – teda sa vypíše chybná sprava a obnoví status charakteru na „End Round“. Ďalej nemôže nič robiť, kým Round neukonči iná postava, status ktorej sa mení na „His Turn“. Do combat_log-u sa zapíše čo sa stalo
2. Ma dostatok current_ap na cast spellu – Systém obnoví množstvo jeho aktuálnych AP bodov. Zoberie z tabuľky jeho target Character údaje *dexterity*, *health*, a z tabuľky jeho Classy *armor_bonus* stáť. Systém hodí kockou d20 a pripočíta to číslo k aktuálnemu *configured_attribute* postavy, ktorá spell castuje. To číslo sa porovná s Armour hodnotou Target postavy, ktorá sa počíta nasledovne:

$$AC = 10 + (target.dexterity / 2) + class.armor\_bonus$$

Ďalej scenár sa rozdelí na ešte 2 možnosti.

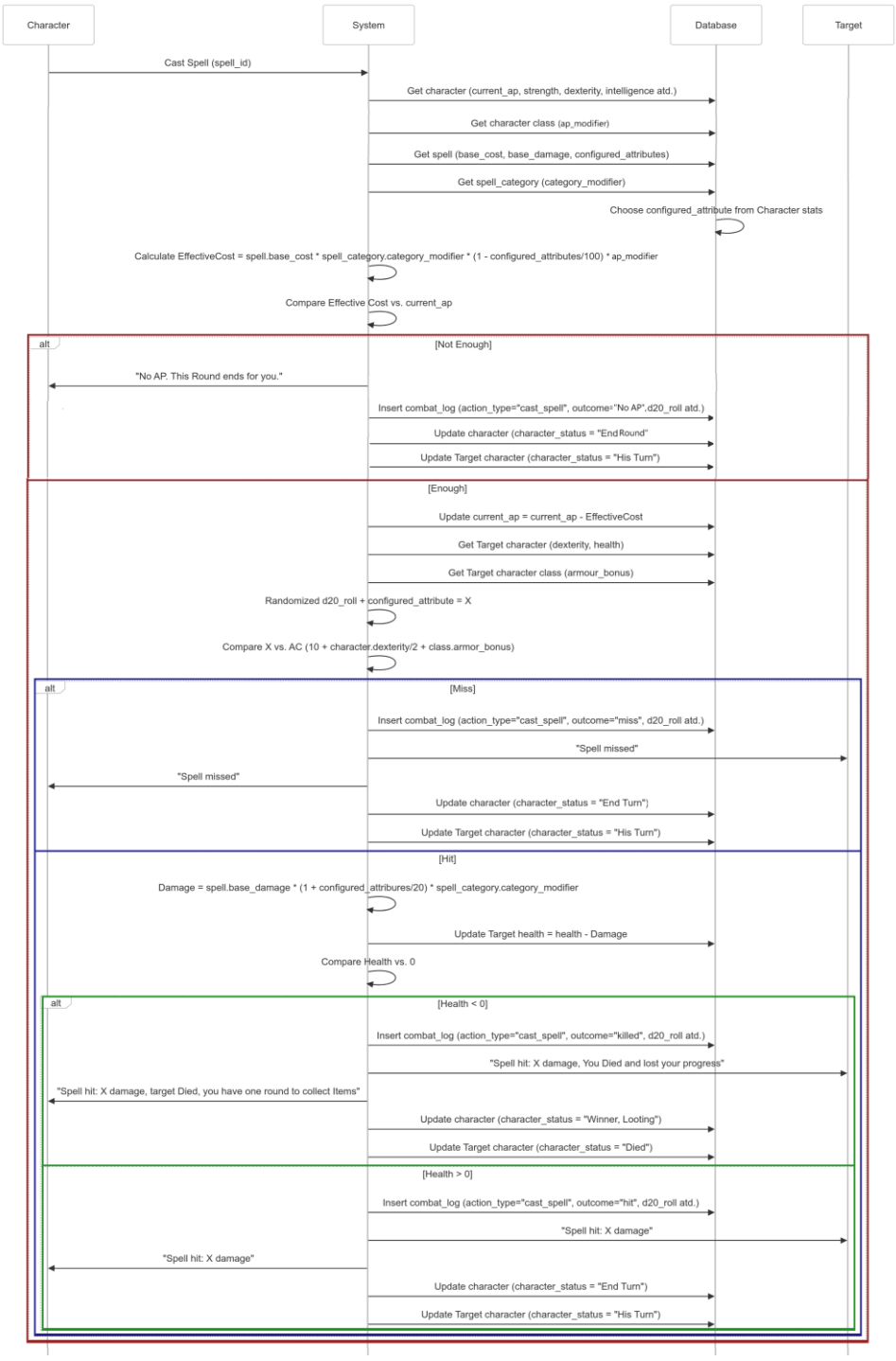
- 2.1 AC target postavy je vyšší, než šťastie prvej postavy. Do databázy sa pridá novy záznam v ktorom sa zapíše čo sa pravé stalo. Postavám prídu charakterne správy a ich charakter statusy sa zmenia na „End Turn“ a „His Turn“.
- 2.2 Charakter ešte ma nejaké šťastie a systém sa rozhodol, že hráč trafil. Vtedy sa vypočíta damage Spellu nasledovanou formulou:

*Damage = spell.base_damage * (1 + configured_attribute_value / 20) * spell_category.category_modifier*

Spravený obrovsky (asi) damage sa odčíta v databáze od HP nepriateľa a systém sa pozrie, ci ešte vôbec ma hp. Ďalej sa to znovu rozdelí na 2 scenáre.

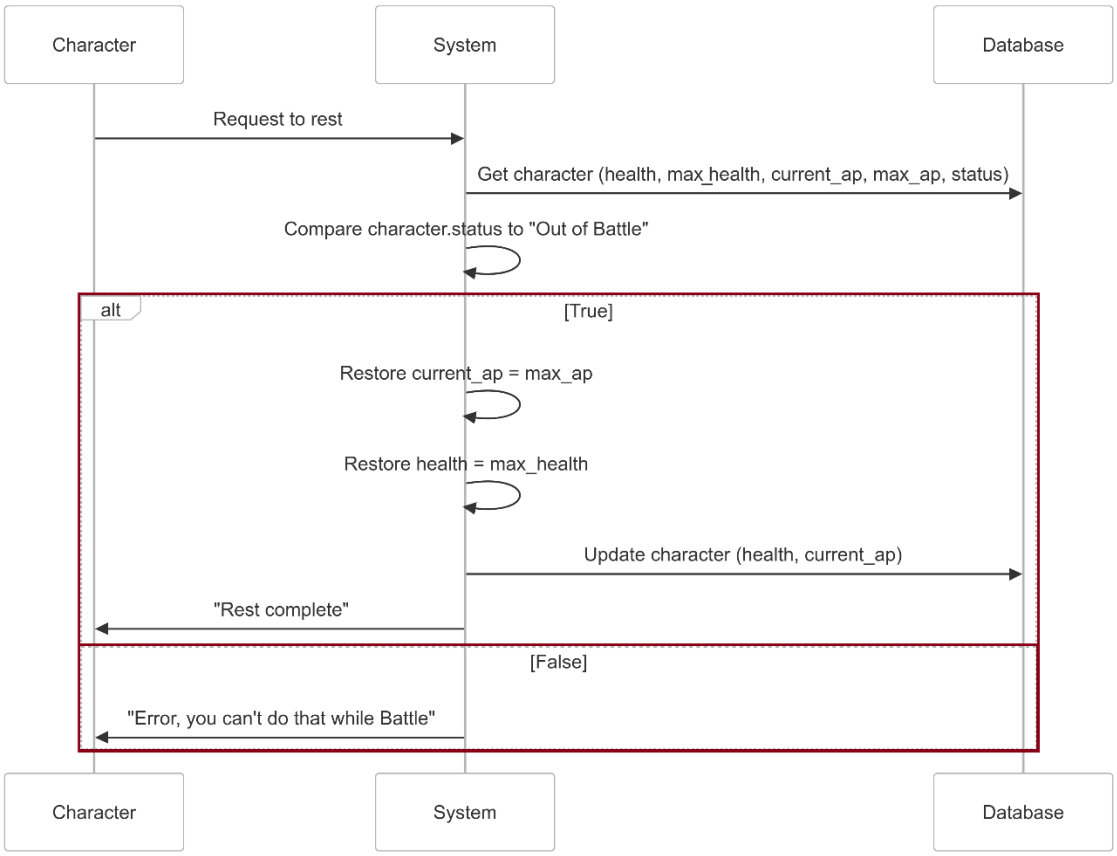
2.2.1 Nepriateľ nemá hp a teda zomiera. To sa zapíše do logu, postavy o tom zistia cez správy a ich status sa zmení. Nás hrdina bude mat status „Winner, looting“ a bude mat cely Round, aby nazbierať všetko čo zle leží. Nepriateľ ale zomrie a už nič nebude môcť, lebo k tomu ešte aj všetko stratil.

2.2.2 Spell neporazil nepriateľa a ten to prežil. To sa tiež zapíše do logu, postavy zistia kto koľko damagu utrpel a statusy Charakterov sa vymenia v databáze na „End Turn“ pre toho, kto spell spravil a „His Turn“ pre toho, kto ten útok prežil.

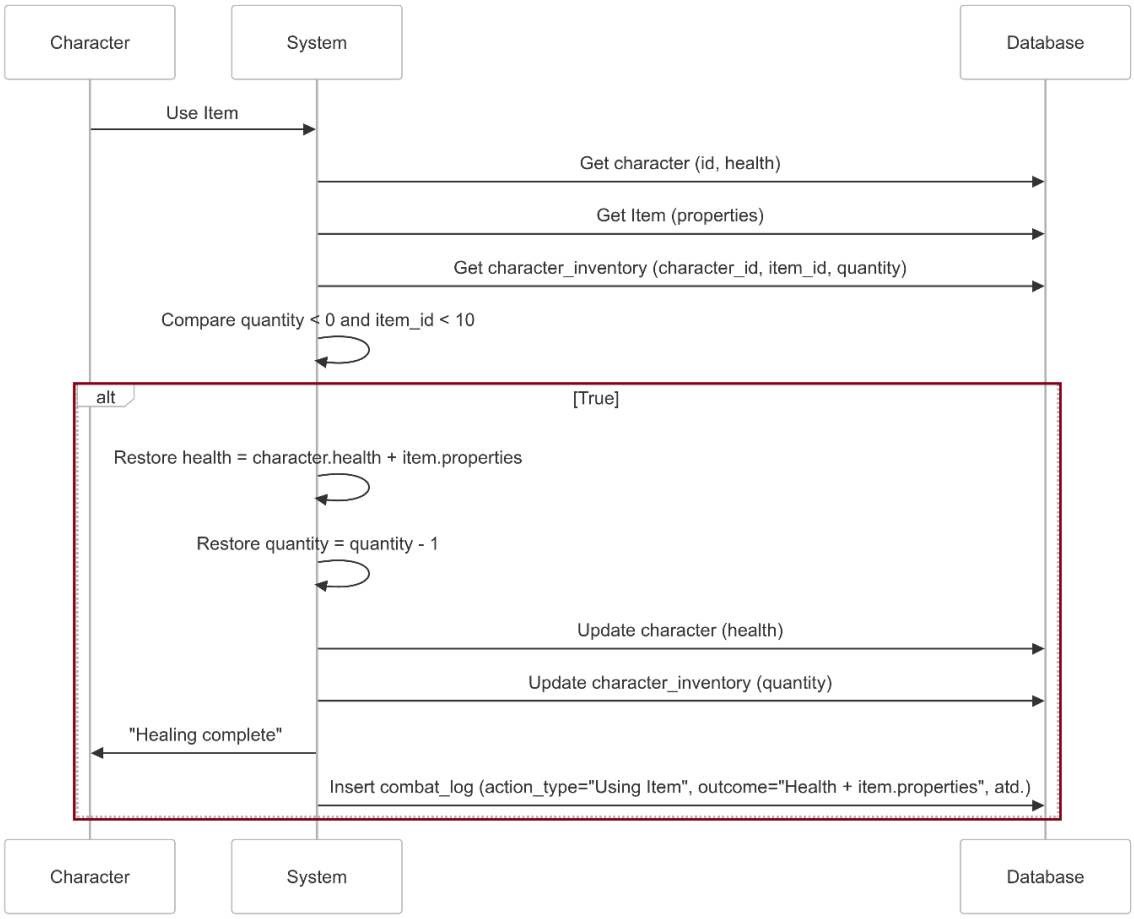


3.2 Oddych a obnovenie zdravia (2 scenár podlá Zadania)

Keď postava žiada oddych a obnovenie zdravia a akčných bodov, systém načíta aktuálne údaje (health, max_health, current_ap, max_ap a status). Ak je status "Out of Battle", hodnoty sa prepisujú na maximálne a uložia sa do databázy. Potom sa vráti potvrdenie. Ak status nie je vhodný, zobrazí sa chybové hlásenie, pretože postava nemôže oddychovať počas bitky.



Postava tak isto môže ale si pridať HP, alebo AP počas bitky pomocou Itemov. Systém dostane požiadavku na použitie itemu. Následne vytiahne údaje z databázy, ako kto chce použiť item a koľko ma zdravia, koľko ma toho itemu v inventári a čo ten item ma robiť. Ak všetko bude úspešne sedieť (v rámci rozoberania iba healing mechaniky de fakto predpokladáme, že áno). Zdravie postavy sa pridá, item sa odčíta z inventáru, to sa uloží do Databasy a zároveň sa vytvorí záznam v combat_log-e, že postava použila heal. V danom prípade nemusíme riešiť ani AP pointy, a ani status postavy. V hre je možnosť použiť item kedykoľvek pred útokom a tým, že mame turn_based systém určujeme poradie kto ma svoj turné – mimo tejto mechaniky. Respektíve – na začiatku bitky a po použití Spellu.

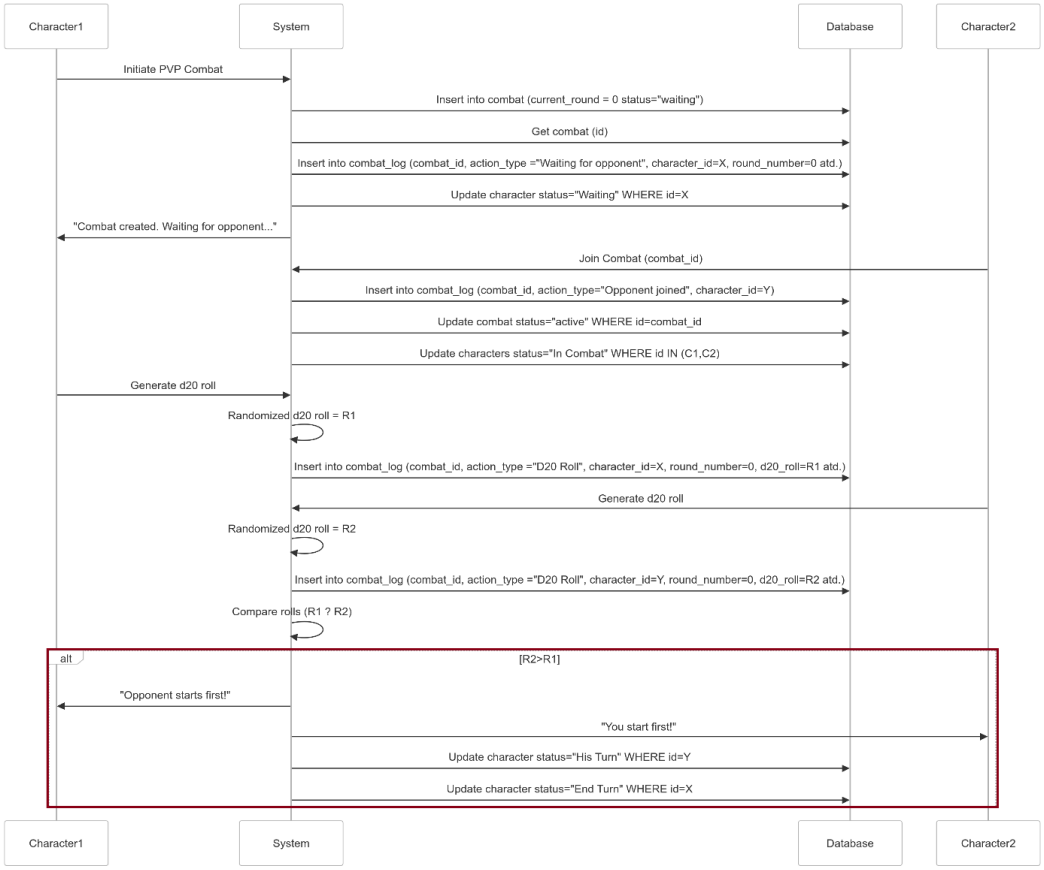


3.3 Iniciácia boja (3 scenár podlá zadania)

Keď postava vytvára vlastne Lobby na PVP – systém vytvorí novy záznam v Combat tabuľke s statusom “waiting” a *current_round*=0. Pri tom berie aktuálne vytvorený PK combat. Následne systém vytvorí záznam v combat_log tabuľke s parametrami *action_type*="Waiting for opponent", link na id charakteru, ktorá vytvorila lobby and *round_number*=0. Tak isto sa zmení aj status character.status na „Waiting“.

Takto budeme pokračovať, kým sa nepripojí character2, na zvolený combat_id, teda nepriateľ. Combat.status sa zmení na „active“ a statusy oboch hráčov na „In Combat“. Odteraz Lobby je zavreté a živý vystúpi iba jeden.

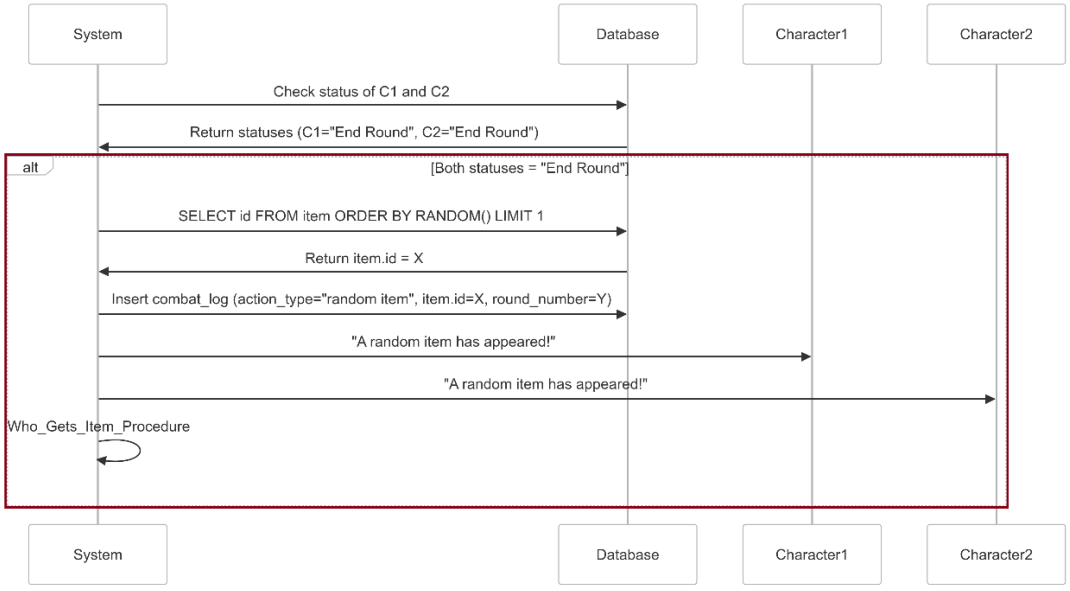
Systém požiada hráčov hodiť kockou. Ten, kto vytvoril Lobby – ma prednosť. Výsledok jeho hodu sa zapíše do combat_log s *action_type*="D20 Roll", *round_number*=0, a hodnotou rollu *d20_roll.* To iste spraví aj character2. Systém porovná dve hodnoty a podlá toho, akému character_id patrilo väčšie číslo - rozhodne kto začína. Spraví to príslušnou zmenou statusov „His Turn“ a „End Turn“.



3.4 Pridanie predmetu do boja (5 scenár podlá zadania)

Keď majú obaja Character1 a Character2 nastavený status na "End Round" v tabuľke characters, systém skontroluje ich stav. Ak je podmienka splnená, vykoná dopyt na tabuľku item pomocou RANDOM() LIMIT 1 dopytu (približne tak sa to bude realizovať), čím náhodne vyberie jeden predmet (napr. item_id=3). Systém potom vloží nový záznam do combat_log s action_type=“random Item“, vybraným item_id a aktuálnym round_number. Obaja hráči dostanú notifikácie.

Nakoniec systém spustí internú procedúru Who_Gets_Item_Procedure, ktorá určí vlastníctvo predmetu podlá d20 kocky.



3.5 Získanie predmetu (6 scenár podlá zadania)

Keď postava skúsi zobrať predmet, systém najprv získa jej atribúty ako strength a constitution (odolnosť) z tabuľky characters. Zároveň dostane inventory_modifier (modifikátor inventára) z tabuľky class. Následne získa aktuálnu váhu inventára postavy, ktorá sa vypočíta zvlášť a ďalej ta vaha + vaha predmetu ktorý chce sa porovná s jeho kapacitou, ktorá sa vypočíta podlá formuly:

*Kapacity = (character.strenght + character.constitution) * class.inventory_modifier*

Ak váha inému splna požiadavku, zoznam character_inventory danej postavy búd sa updatne, alebo upraví podlá toho, či dostal novy item, alebo nejaký, čo už ma. Tak isto sa vytvorí záznam v combat_log ohľadom toho. V prípade neúspechu vypíše chybu postave a aj v combat_log-e