

# Protokol k zadaniu 3

## *Modelovanie navrhnutej databázy*

Predmet: Databázové systémy  
2024/2025

**Johanna Tilešová**

Slovenská technická univerzita v Bratislave  
Fakulta informatiky a informačných technológií

`xtilesova@stuba.sk`

27. apríl 2025



SLOVENSKÁ TECHNICKÁ  
UNIVERZITA V BRATISLAVE  
FAKULTA INFORMATIKY  
A INFORMAČNÝCH TECHNOLOGIÍ

### **Abstrakt**

Táto práca dokumentuje návrh relačnej databázy pre systém na sledovanie a spracovanie bojových interakcií v prostredí inšpirovanom RPG hrami. Cieľom bolo vytvoriť komplexný model postáv, kúziel, predmetov a bojových situácií s dôrazom na flexibilitu, modularitu a analytickú silu systému. Dokumentácia zahŕňa opis fiktívneho herného sveta, návrh entít, atribútov a ich vzťahov, ako aj logický a konceptuálny model databázy. V druhej časti sú popísané kľúčové herné procesy, ako napríklad zoslanie kúzla, vstup do boja či manipulácia s inventárom. Výsledkom je plnohodnotný návrh databázovej štruktúry, ktorá je pripravená na implementáciu alebo rozšírenie o hernú logiku.

# Obsah

<b>1</b>	<b>Logicko-fyzické mapovanie modelu</b>	<b>4</b>
<b>2</b>	<b>Rozdiely oproti pôvodnému návrhu</b>	<b>5</b>
<b>3</b>	<b>Procesné toky a prototypy postupov</b>	<b>5</b>
3.1	sp_rest_character . . . . .	5
3.2	sp_cast_spell . . . . .	5
3.3	sp_enter_combat . . . . .	6
3.4	sp_loot_item . . . . .	7
3.5	sp_swap_items . . . . .	8
<b>4</b>	<b>Funkcie na výpočty</b>	<b>8</b>
4.1	f_effective_spell_cost . . . . .	8
4.2	f_calculate_armor_class . . . . .	9
4.3	f_calculate_spell_damage . . . . .	9
4.4	f_current_inventory_weight . . . . .	9
4.5	f_max_inventory_capacity . . . . .	9
4.6	f_roll_d20 . . . . .	10
<b>5</b>	<b>Zoznam vytvorených indexov</b>	<b>10</b>
<b>6</b>	<b>Testovanie</b>	<b>10</b>
6.1	Combat Simulation . . . . .	11
6.2	Full Inventory . . . . .	12
6.3	Swap Item . . . . .	12
6.4	Attack . . . . .	12
6.5	Healing Spell . . . . .	13
6.6	Čistenie testovacích dát . . . . .	13
<b>7</b>	<b>Pokyny na testovanie</b>	<b>13</b>



## 2 Rozdiely oproti pôvodnému návrhu

Logický model bol zachovaný v súlade so zadáním 2. Jedinou zmenou bolo odstránenie tabuľky `attributes`, ktorá pôvodne obsahovala iba názvy atribútov (napr. `strength`, `dexterity`, `intelligence` a podobne). V súčasnosti sú názvy aj hodnoty atribútov spravované spoločne v tabuľke `character_attributes`, ktorá obsahuje stĺpce `attribute_name` a `value`. Týmto spôsobom som model mierne zjednodušila, keďže bolo neefektívne udržiavať samostatnú tabuľku výhradne pre názvy atribútov.

## 3 Procesné toky a prototypy postupov

Procesné toky predstavujú hlavné herné mechanizmy, ktoré boli implementované v databázovom modeli prostredníctvom procedúr a funkcií. Tieto procesy definujú, akým spôsobom postavy interagujú s herným prostredím, funkcionality akcií v boji, manipuláciu s inventárom a rôzne dynamické zmeny v stave postáv.

### 3.1 `sp_rest_character`

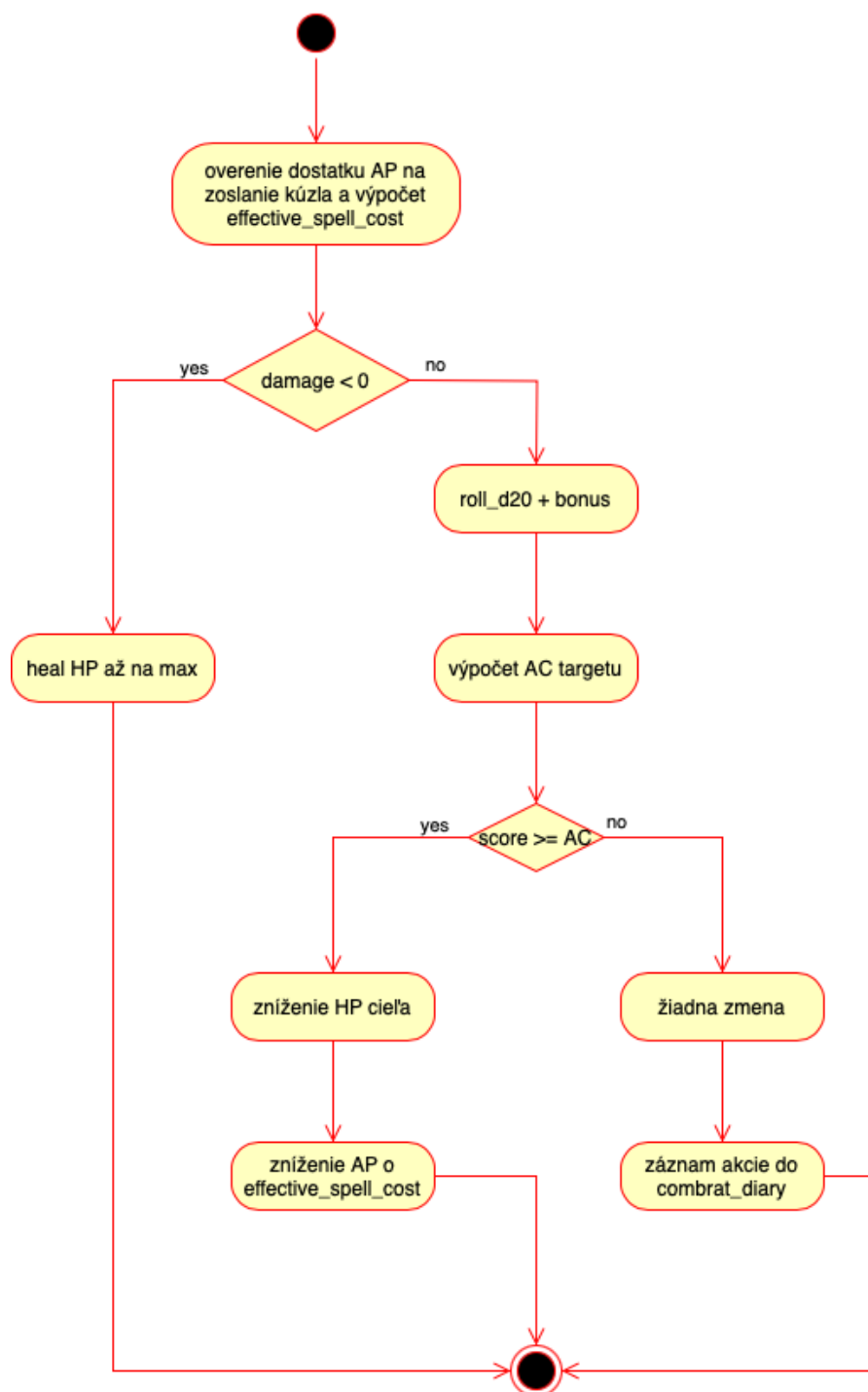
Procedúra `sp_rest_character` umožňuje postave oddychovať mimo aktívneho boja, čím si obnovuje svoje zdravie (HP) a akčné body (AP).

- **Krok 1:** Overí sa, či existuje postava so zadaným ID a či sa momentálne nenachádza v boji (`in_combat = false`).
- **Krok 2:** Ak je podmienka splnená, obnoví sa hodnota `current_hp` postavy na jej maximum `max_hp`.
- **Krok 3:** Skontroluje sa, či sa postava nachádza v tabuľke `combat_participants` (napríklad mohla byť pripravená na budúci boj alebo bola v minulosti v boji).
- **Krok 4:** Ak je postava v tabuľke `combat_participants`, obnoví sa jej aktuálny počet akčných bodov (`current_ap`) na maximálnu hodnotu (`max_ap`), ale iba v prípade, že súčasná hodnota `current_ap` je nižšia ako `max_ap`.

Procedúra zabezpečuje, aby sa postava mohla pred novým bojom plne zotaviť a bola pripravená na ďalšie akcie.

### 3.2 `sp_cast_spell`

Procedúra `sp_cast_spell` simuluje zoslanie kúzla postavou na cieľ v rámci aktívneho boja. Postup overuje akčné body, vykonáva hod kockou d20 na určenie zásahu, vypočítava poškodenie a zaznamenáva udalosť. (Obrázok 2)



Obr. 2: Aktivitý diagram pre procedúru sp\_cast\_spell

### 3.3 sp\_enter\_combat

Procedúra sp\_enter\_combat pridá postavu do aktívneho boja a nastaví jej základné bojové parametre.

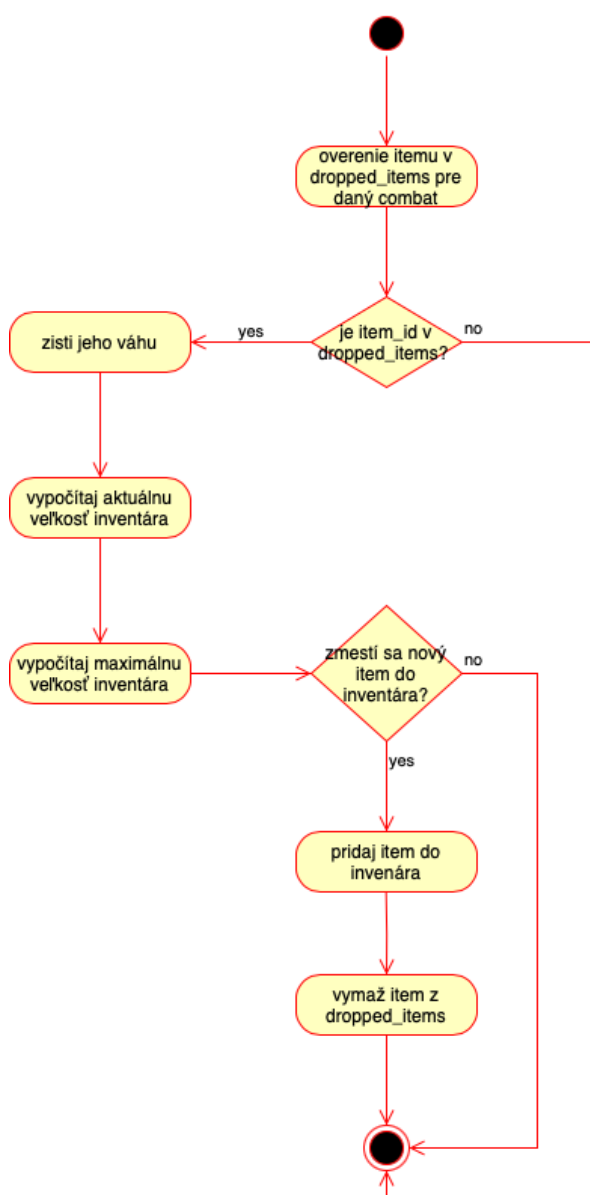
Najprv sa zistí maximálny počet akčných bodov (`max_ap`) pre postavu. Následne sa vypočíta nové poradie iniciatívy ako maximum existujúcich hodnôt v boji (`initiative_order`)

plus jedna. Postava sa vloží do tabuľky `combat_participants` s aktuálnym počtom akčných bodov, stavom 'alive' a priradeným poradím.

Nakoniec sa v tabuľke `characters` aktualizuje príznak `in_combat` na `true`, čím sa označí, že postava je v boji.

### 3.4 sp\_loot\_item

Procedúra `sp_loot_item` spracováva proces získania predmetu (*loot*) postavou z bojovej oblasti. Zabezpečuje kontrolu inventárnych limitov postavy a následnú aktualizáciu vlastníctva predmetu. (Obrázok 3)



Obr. 3: Výsledok testu `t_attack`

### 3.5 sp\_swap\_items

Jedna custom procedúra, ktorá je odvodená od procedúry pre plný inventár, ale rieši výmenu itemov.

Procedúra `sp_swap_items` umožňuje postave vymeniť predmet zo svojho inventára za iný predmet z bojovej oblasti v prípade, že už nemá v inventári dostatok miesta. Táto mechanika simuluje realistické správanie, kde hráč musí obetovať existujúci predmet, aby mohol získať nový.

- **Krok 1:** Overí sa váha predmetu, ktorý chce postava zahodiť (`p_item_id_drop`) a predmetu, ktorý chce získať (`p_item_id_pick`).
- **Krok 2:** Skontroluje sa, či má predmet na zahodenie váhu rovnakú alebo vyššiu než predmet, ktorý chce postava získať. Ak nie, procedúra vyvolá výnimku.
- **Krok 3:** Predmet na zahodenie sa odstráni z inventára postavy. Ak jeho množstvo klesne na nulu, úplne sa vymaže z inventára.
- **Krok 4:** Predmet na zahodenie sa vloží do tabuľky `dropped_items`, čím sa stáva súčasťou herného prostredia.
- **Krok 5:** Vypočíta sa aktuálna váha inventára (`f_current_inventory_weight`) a porovná sa s maximálnou kapacitou (`f_max_inventory_capacity`).
- **Krok 6:** Ak má postava po výmene dostatok miesta v inventári:
  - Nový predmet sa pridá do inventára postavy.
  - Nový predmet sa odstráni z tabuľky `dropped_items`.
- **Krok 7:** Ak ani po výmene nie je dostatok miesta, procedúra vyvolá výnimku a operáciu neuskutoční.

Procedúra zabezpečuje dynamickú správu inventára a podporuje rozhodovanie hráča v situáciách s limitovaným miestom na predmety.

## 4 Funkcie na výpočty

V tejto časti sú predstavené viaceré pomocné funkcie, ktoré sú integrované do procedúr s cieľom zjednodušiť výpočty a zabezpečiť lepšiu prehľadnosť kódu. Základnou požadovanou funkciou podľa zadania bol výpočet `f_effective_spell_cost`. Pre správnu a efektívnu realizáciu všetkých procedúr som následne doplnila aj niekoľko vlastných (custom) funkcií, aby sa výpočty nemuseli realizovať priamo v rámci procedúr.

### 4.1 f\_effective\_spell\_cost

Funkcia `f_effective_spell_cost` vypočíta efektívnu cenu kúzla (`AP cost`) pre danú postavu. Výsledná cena závisí od základnej ceny kúzla a hodnoty špecifického atribútu postavy, pričom sa berie do úvahy aj multiplikátor vplyvu atribútu. Výpočet prebieha podľa vzorca:

$$\text{EffectiveCost} = \text{BaseCost} \times (1 \pm (\text{AttributeValue} \times \text{Multiplier})),$$



kde znak  $\pm$  závisí od typu modifikátora (increase alebo decrease). Funkcia zabezpečuje, že výsledná cena nikdy neklesne pod hodnotu 1.

## 4.2 f\_calculate\_armor\_class

Funkcia `f_calculate_armor_class` slúži na výpočet hodnoty brnenia (*Armor Class*, AC) pre zadanú postavu. Výsledná hodnota AC ovplyvňuje pravdepodobnosť zásahu v boji a je vypočítaná podľa vzorca:

$$\text{ArmorClass} = 10 + \left( \frac{\text{Dexterity}}{2} \right) + \text{ClassArmorBonus}.$$

Pri výpočte funkcia:

- Získa hodnotu atribútu `dexterity` pre danú postavu z tabuľky `character_attributes`.
- Zistí bonus brnenia triedy postavy (`class_armor_bonus`) z tabuľky `class`.
- Spočíta výslednú hodnotu AC podľa vzorca.

Funkcia je kľúčová pre bojové mechaniky hry, keďže určuje, aké ťažké je pre protivníkov zasiahnuť postavu.

## 4.3 f\_calculate\_spell\_damage

Funkcia `f_calculate_spell_damage` vypočíta výsledné poškodenie kúzla (damage) zoslaného postavou. Výsledná hodnota závisí od základného poškodenia kúzla a hodnoty špecifického atribútu kúzelníka, podľa vzorca:

$$\text{EffectiveDamage} = \text{BaseDamage} \times \left( 1 + \frac{\text{ConfiguredAttribute}}{20} \right).$$

Ak kúzlo nemá nakonfigurovaný atribút pre poškodenie, použije sa iba základná hodnota poškodenia. Funkcia zabezpečuje dynamickú škálovateľnosť účinnosti kúziel podľa sily postavy.

## 4.4 f\_current\_inventory\_weight

Funkcia `f_current_inventory_weight` vypočíta aktuálnu hmotnosť inventára postavy. Súčet sa počíta ako váha každého predmetu vynásobená jeho množstvom v inventári:

$$\text{TotalWeight} = \sum (\text{ItemWeight} \times \text{Quantity})$$

## 4.5 f\_max\_inventory\_capacity

Funkcia `f_max_inventory_capacity` vypočíta maximálnu nosnosť inventára postavy. Výsledná hodnota závisí od súčtu atribútov `Strength` a `Constitution`, upraveného koeficientom triedy (`inventory_modifier`):

$$\text{MaxCapacity} = (\text{Strength} + \text{Constitution}) \times \text{InventoryModifier}$$

## 4.6 f\_roll\_d20

Funkcia `f_roll_d20` simuluje náhodný hod dvadsaťstennou kockou (d20). Výsledkom funkcie je celé číslo v rozsahu od 1 do 20, generované pomocou pseudonáhodnej funkcie.

Funkcia sa využíva najmä pri určovaní zásahov v boji.

## 5 Zoznam vytvorených indexov

V rámci optimalizácie výkonu boli vytvorené indexy na často používané cudzie kľúče a stĺpce využívané pri JOIN operáciách a filtrovaní údajov.

Indexy na stĺpcoch ako `character_id`, `item_id` a `combat_id` v tabuľkách `inventory`, `dropped_items` a `combat_participants` zlepšujú rýchlosť spojovaní medzi tabuľkami počas bežných dotazov. Tieto stĺpce sú kľúčové pre herné mechaniky ako je manipulácia s inventárom, záznamy v boji alebo správa voľných predmetov na bojisku.

Ďalšie indexy boli vytvorené na najčastejšie filtrované stĺpce v podmienkach `WHERE`:

- `status` v tabuľke `combat_participants` pre rýchle vyhľadanie živých alebo mŕtvych postáv,
- `in_combat` v tabuľke `characters` na kontrolu účasti v boji,
- `quantity` v tabuľke `dropped_items` na efektívne spracovanie dostupnosti predmetov.

Týmto nastavením indexov sa výrazne zrýchľuje vykonávanie dotazov v kritických častiach systému, najmä pri boji, loote a manipulácii s inventárom.

## 6 Testovanie

Pred testovaním som si pripravila vlastné testovacie dáta a scenáre, aby som pokryla rôzne *edge cases* a situácie v boji aj pri manipulácii s inventárom. Vytvorila som viacero postáv so špecifickými atribútmi: `TestSwapHero` na výmenu predmetov, `TestCasterHero` so zameraním na kúzla, `TestTankHero` na tankovanie, `TestMageHero` na agilitu a `TestWarriorHero` na fyzické útoky. Každá postava mala nakonfigurovaný inventár rôznej váhy a na bojisku boli rozmiestnené aj predmety (`dropped_items`).

Táto príprava umožnila efektívne otestovať všetky hlavné mechaniky systému.

## 6.1 Combat Simulation

Test `combat_simulation` overuje kompletný priebeh boja medzi viacerými postavami. Po vytvorení boja sa do neho pridajú všetky testovacie postavy. Boj je rozdelený na kolá (rounds), v rámci ktorých postavy náhodne útočia pomocou kúziel.

Výpočet zásahu zahŕňa hod kockou (`f_roll_d20`), bonus z atribútu a porovnanie s `Armor Class` cieľa. Každý zásah alebo neúspech je zaznamenaný v `combat_diary`. Postava, ktorá stratí všetky HP, je vyradená a jej predmety sa presunú na bojisko.

Boj pokračuje, až kým neostane len jedna živá postava. Test overuje fungovanie útokov, manažment akčných bodov, úmrtí aj správne ukončenie boja. (Obrázok 4)

```
-----
🔄 Starting Round 1
🎯 TestTankHero (id 18 | HP: 5 | AP: 5) attacks TestMageHero (id 19 | HP: 0 | AP: 8)
⚡ Not enough AP to cast Lightning Strike, skipping.
🎯 TestWarriorHero (id 20 | HP: 0 | AP: 6) attacks TestSwapHero (id 16 | HP: 0 | AP: 6)
❌ Miss!
💀 TestSwapHero (id 16) died and dropped their loot!
🎯 TestTankHero (id 18 | HP: 5 | AP: 5) attacks TestMageHero (id 19 | HP: 0 | AP: 8)
⚡ Not enough AP to cast Lightning Strike, skipping.
🎯 TestMageHero (id 19 | HP: 0 | AP: 8) attacks TestWarriorHero (id 20 | HP: 0 | AP: 1)
✅ Hit! Damage: 25
💀 TestWarriorHero (id 20) died and dropped their loot!
🎯 TestTankHero (id 18 | HP: 5 | AP: 5) attacks TestMageHero (id 19 | HP: 0 | AP: 2)
⚡ Not enough AP to cast Lightning Strike, skipping.
🎯 TestTankHero (id 18 | HP: 5 | AP: 5) attacks TestMageHero (id 19 | HP: 0 | AP: 2)
❌ Miss!
💀 TestMageHero (id 19) died and dropped their loot!
🏆 Combat finished. One survivor!
```

Obr. 4: Výsledok testu `t_combat_simulation`

## 6.2 Full Inventory

Tento test overuje, ako systém reaguje pri pokuse o loot predmetu v závislosti od kapacity inventára. Ak má postava dostatok miesta, predmet je pridaný. Ak kapacitu prekračuje, lo-otovanie zlyhá s chybovou hláškou. Test pokrýva obidve situácie a overuje správne uplatnenie obmedzenia nosnosti.

## 6.3 Swap Item

Test `swap_item` simuluje výmenu predmetu pri plnom inventári. Postava `TestSwapHero` sa pokúsi lootnúť predmet z bojiska. Ak nemá miesto, systém automaticky vyhadzuje najľahšie predmety, kým sa neuvoľní dostatok kapacity.

Po výmene sa kontroluje, či je nový predmet správne uložený v inventári, čím sa overuje fungovanie swapovania aj v hraničných situáciách. (Obrázok 5)

```
🎯 Selected item to pick: ID 79, Weight 20
📦 Character inventory:
Max capacity: 60
Current weight: 51
Remaining capacity: 9
🔄 Loop: current_weight=51, max_capacity=60, item_weight=20
🗑 Dropping item ID 78 with weight 2
🔄 Loop: current_weight=49, max_capacity=60, item_weight=20
🗑 Dropping item ID 80 with weight 5
🔄 Loop: current_weight=44, max_capacity=60, item_weight=20
🗑 Dropping item ID 81 with weight 8
🔄 Loop: current_weight=36, max_capacity=60, item_weight=20
✅ Test passed: New item correctly looted after swap.
```

Obr. 5: Výsledok testu `t_swap_items`

## 6.4 Attack

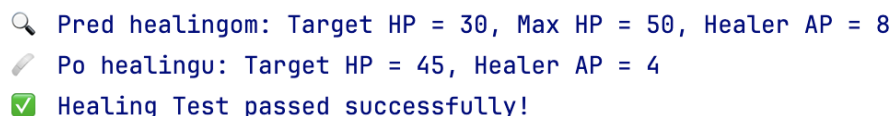
Test `attack` overuje použitie útočného kúzla počas boja. Náhodne sa vyberú dve živé postavy, pričom jedna vykoná útok pomocou náhodného kúzla. Test kontroluje dostatok akčných bodov, hod kockou (`f_roll_d20`), výpočet útoku a úspešnosť zásahu podľa `Armor Class` cieľa. (Obrázok 6)

```
🧙 Použité kúzlo: "Lightning Strike", ovplyvnené atribútom "intelligence"
🎯 Útočník ID: 1, Cieľ ID: 2
🎲 Hod kockou: 2, Útočný bonus: 12, Armor Class cieľa: 20
⚡ Útočné skóre: 14 (roll + bonus) vs AC: 20
❌ Útok neúspešný! (miss)
```

Obr. 6: Výsledok testu `t_attack`

## 6.5 Healing Spell

Test `healing_spell` overuje fungovanie liečivého kúzla počas boja. Vyberie sa zranená postava a ako liečiteľ je použitý `TestCasterHero` s kúzlom `Healing Mist`. Test kontroluje dostatok akčných bodov, správne zvýšenie `current_hp` a neprekročenie maximálneho zdravia (`max_hp`). (Obrázok 7)



The image shows a terminal window with three lines of text. The first line is preceded by a magnifying glass icon, the second by a pencil icon, and the third by a green checkmark icon.

```
Pred healingom: Target HP = 30, Max HP = 50, Healer AP = 8
Po healingu: Target HP = 45, Healer AP = 4
Healing Test passed successfully!
```

Obr. 7: Výsledok testu `t_healing_spell`

## 6.6 Čistenie testovacích dát

Po ukončení testovania bol spustený čistiaci skript, ktorý vymazal všetky testovacie postavy, predmety, záznamy o boji a inventári. Skript zároveň odstránil prázdne bojové relácie a resetoval ID sekvencie pre ďalšie správne testovanie.

## 7 Pokyny na testovanie

Pred samotným spustením testov je potrebné najprv vytvoriť testovacie dáta pomocou skriptu `t_test_data`. Po načítaní vzorových údajov je možné vykonať simuláciu boja cez skript `combat_simulation`, ktorý vytvorí aktívnu bojovú reláciu medzi viacerými postavami.

Po skončení boja zostanú na bojisku voľné predmety, vďaka čomu je možné následne spustiť testy zamerané na manipuláciu s inventárom, ako napríklad `full_inventory` alebo `swap_item`. Následne je možné otestovať aj ďalšie herné mechaniky, ako napríklad útok prostredníctvom skriptu `attack` alebo liečenie pomocou `healing_spell`.

Tento postup zabezpečí, že všetky testovacie scenáre budú mať k dispozícii potrebné podmienky na správne vykonanie a overenie funkcionality systému.