SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Hrvoje Lesar

IZRADA PLATFORME ZA PRAĆENJE BITKE U IGRI IGRANJA ULOGA

DIPLOMSKI RAD

Varaždin, 2025.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Hrvoje Lesar

Matični broj: 0016133479

Studij: Organizacija poslovnih sustava

IZRADA PLATFORME ZA PRAĆENJE BITKE U IGRI IGRANJA ULOGA

DIPLOMSKI RAD

Mentor:

Prof. dr. sc. Markus Schatten

Varaždin, rujan 2025.

Hrvoje Lesar

Izjava o izvornosti

Izjavljujem da je ovaj diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor potvrdio prihvaćanjem odredbi u sustavu FOI Radovi

Sažetak

Opsega od 100 do 300 riječi. Sažetak upućuje na temu rada, ukratko se iznosi čime se rad bavi, teorijsko-metodološka polazišta, glavne teze i smjer rada te zaključci.

Ključne riječi: riječ; riječ; ...riječ; Obuhvaća 7+/-2 ključna pojma koji su glavni predmet rasprave u radu.

Sadržaj

1. Uvod

- Rpg
- Cilj
- Prvi dio...
- Praktični dio

2. Igre igranja uloga

Igre igranja uloga predstavljaju jedan od najsloženijih i kreativnih spojeva priče, izvedbe i igre u suvremenoj kulturi. Ovakva igra poziva igrače da zajednički stvaraju i nastanjuju zamišljeni svijet, vođeni kombinacijom strukturiranih pravila i spontanog pripovijedanja. Svaka sesija je istovremeno i igra i priča, a njezin ishod oblikuju mašta igrača i slučajnost ishoda bacanja kockica.

Igre igranja uloga nemaju jednu definiciju i možemo ih definirati s različitih gledišta:

- Situacija igranja uloga definirana kao situacija u kojoj se od pojedinca izričito traži da preuzme ulogu koja inače nije njegova, ili ako jest njegova, onda u okruženju koje nije uobičajeno za izvođenje te uloge. [1]
- Igranje uloga nije jedinstvena jasno definirana aktivnost, već čitav niz aktivnosti okupljenih pod pogodnim nazivom. Na jednom kraju spektra nalazi se intenzivno odražavanje sobnih emocija, dok se na drugom kraju nalazi situacija u kojoj je preuzimanje uloge bliže konceptu zagovaranja. [2]
- Igranje uloga je umjetnost iskustva, a stvaranje igra uloga znači kreiranje novih iskustava. [2]
- Igranje uloga definira se kao bilo koji čin u kojem se istovremeno stvara, dodaje i promatra imaginarna stvarnost. [2]
- Igra igranja uloga mora se sastojati od interaktivnog pripovijedanja: sposobnosti likova i razrješenja radnji, definirani su brojevima ili količinama, kojima se manipulira prema određenim pravilima. Donošenje odluka igrača pokreće i pomiče priču u naprijed. Uz skupinu koja djeluje kao autor, priča organski raste i odigrava se, bivajući doživljena od svojih stvaratelja. [2]

2.1. Ratne igre

Početak i inspiracija za igre igranja uloga dolazi iz ratnih igara. Nasljeđe ratnih igara seže od šaha, budući da su prve igre unutar posebne kategorije ratnih igara uvelike posuđivale ploče, figure i mehanike upravo iz šaha. Georg von Reißwitz se smarta ocem ratnih igara, jer je razvio prvi sustav ratnih igara koji je široko korišten ako ozbiljan alat za obuku i istraživanje. Razvijenu igru su nazvali tzv. `Das Kriegsspiel` te je igra zadovoljila dugo prepoznatu potrebu za jeftinim i lako ponovljivim sredstvom za obuku časnika u zapovijedanju i planiranju bitke. Vojne ratne igre gotovo se uvijek fokusiranju na sadašnjost, na ondašnje vojske, tehnologiju i države u vrijeme njihova nastanka. Veliki dio razvoja ratnih igara osamnaestog i devetnaestog stoljeća održava neprestana poboljšanja u sredstvima i provedbi ratovanja te posljedičnu potrebu da se ratne igre stalno usklađuju s realnošću. [3]

Tek kad su se hobisti počeli poigravati ovim sustavima, uspjeli su osloboditi ratne igre ograničenja suvremenog konteksta i istražiti povijesna razdoblja, buduće moguće svjetove, pa

čak i nemoguće fantastične svjetove. Hobisti su također odbacili strogo reproduciranje stvarnih uvjeta na bojištu u korist više uravnoteženijeg pristupa koji je kombinirao realističnost i igrivost. Do 1960-ih, ovi zaigrani ljubitelji ratnih igara transformirali su ih iz sredstva za vojnu obuku u znatno maštovitiju aktivnost, onu koja je mogla poslužiti kao temelj za modeliranje događaja u igri poput Dungeons & Dragons (skraćeno D&D). [3]

2.2. Nastanak sustava Dungeons & Dragons

Dungeons & Dragons (kraće poznato kao D&D), igra nastala 1974. godine predstavlja ključni trenutak u povijesti igra uloga. Igru su razvili Gary Gygax i Dave Arneson, koji su težili proširenju mogućnosti stolnih ratnih igara prema novim narativnim i imaginativnim domenama. Prije D&D-a, Gygax je već imao utjecaj u wargaming zajednici, osobito kroz svoj rad na igri `Chainmail`, srednjovjekovnoj miniaturskoj ratnoj igri koju je razvio i bio jedan od autora 1971. godine. `Chainmail` je izvorno zamišljen kao skup pravila za simulaciju srednjovjekovnih bitaka s minijaturama, no uključivao je i fantastični dodatak koji je omogućava igračima da u igru unesu mistična bića i čarobne elemente. Ova dopuna održavala je Gygaxovu fascinaciju srednjovjekovnom literaturom i fantastičnim narativima, posebno djelima J.R.R Tolkeina i Roberta E. Howarda. [4]

Dave Arneson je eksperimentirao s narativnim pristupom u svojoj kampanji Blackmoor, u kojoj su se pojedinačni likovi mogli razvijati kroz više sesija i aktivno utjecati na tijek priče, za razliku od tradicionalnog wargaminga u koje igrači kontroliraju cijele jedinice ili vojske. Arnesonove inovacije, u kombinaciji s Gygaxovim strukturiranim sustavom pravila iz Chainmaila, stvorile su temelje za mehaniku i narativni potencijal igre D&D. [4]

Prvo izdanje D&D-a je izdano 1974. godine, bilo je u početku skromno po opsegu, no uvelo je revolucionarne koncepte; igrači su mogli preuzeti uloge različitih likova istraživati fantastične tamnice, sudjelovati u zadacima vođenim od strane Dungeon Mastera (DM-a), koji moderira svijet igre. Pravila D&D-a uključivala su elemente miniaturskog wargaminga, vjerojatnosnih mehanika s kockama i suradničkog pripovijedanja, stvarajući novi oblik interaktivne zabave koji je spajao strategiju, maštu i narativ. Tijekom vremena igra se razvila kroz više izdanja, od kojih je svako unaprijedilo pravila i proširilo mogućnosti, no osnovni princip suradničkog pripovijedanja i igre vođene likovima je ostao nepromijenjen.

3. Opis pravila igre - Dungeons & Dragons

U ovome poglavlju će biti opisana pravila igre D&D-a. Pravila igre su se razvila od početne koncepcije igre 70-tih godina 20. stoljeća. Poglavlje će biti usredotočeno na pravila definirana u `Player's Handbook 2014`. Svakim novim izdanjem igre pravila se suptilno nagorađuju, poboljšavaju i razvijaju, te se dodaju nove mehanike igre. Ovdje definiramo zadnje izdanje koje će se poglavlje i kasnija implementacije koristiti i pratiti.

Dungeons & Dragons je igra igranja uloga posvećena pripovijedanju u svjetovima mača i čarolije. Dijeli elemente s dječjim igrama pretvaranja. Kao i te igre, D&D pokreće mašta. Radi se o zamišljanju visokog dvorca pod olujnim noćnim nebom i predočavanju kako bi pustolov mogao reagirati na izazove koje taj prizor predstavlja. Za razliku od igra pretvaranja, D&D pričama daje strukturu, način određivanja posljedica za akcije igrača. Igrači bacaju kockice kako bi razriješili jesu li njihovi napadi pogodili ili promašili, ili mogu li se njihovi pustolovi popeti na liticu, izmaknuti se udaru čarobne munje ili izvesti neki drugi opasan pothvat. Sve je moguće, ali kockice čine neke ishode vjerojatnijima od drugih. [5]

Jedan igrač preuzima ulogu Dungeon Mastera, glavnog pripovjedača i suca igre. DM stvara pustolovine za likove, koji se kreću kroz njihove opasnosti i odlučuje koje će putove istražiti. Više o DM-u će biti u potpoglavlju 3.2.

Svaki igrač stvara pustolova (koji se naziva i likom) i udružuje se s drugim pustolovima. Radeći zajedno, grupa može istraživati mračnu tamnicu, ukleti dvorac, izgubljeni hram... Pustolovi mogu rješavati zagonetke, razgovarati s drugim likovima, boriti se protiv čudovišta i otkrivati čudesne čarobne predmete i blago.

3.1. Lista pojmova

Ovo poglavlje će služiti za definiranje terminologije i kratica koje će se koristiti kroz rad. Svaki termin će biti kraće opisan i potencijalno imati engleski naziv (koji je uglavnom više prepoznatljiv nego hrvatski prijevod), no cijeli kontekst ne mora biti razumljiv u spisu, već će biti opisan u nekom od sljedećih poglavlja.

- **D&D** Dungeons & Dragons
- **DM** Dungeon Master
- **NPC** Non-Player Character
- **D20** Kockica s dvadeset strana
- **D2, D4, D6, D8, D10, D12** Kockice s određenim brojem strana
- **AC** Armour class, klasa oklopa
- **DC** Difficulty class, težina zadatka

- **HP** Hit points, životni bodovi
- **XP** Experience points
- **Ability score** Sposobnosti
- **Hit die** Kockica za životne bodove

3.2. Dungeon Master

Dungeon master je kreativna sila iza D&D sesije. DM stvara svijet koji ostali igrači istražuju, te također osmišljava i vodi pustolovine koje pokreću priču. Pustolovina se obično temelji na uspješnom dovršetku neke potrage i može trajati kratko poput jedne sesije ili se protezati kroz više sesija. Duže pustolovine mogu uvući igrače u velike sukobe za čije je rješavanje potrebno više sesija. Kada se povežu u niz pustolovine čine trajnu priču nazvanu kampanja. D&D kampanja može uključivati desetke pustolovina i trajati jednu sesiju, mjesecima ili pa ćak i godinama. [6]

Dungeon master preuzima mnogo različitih uloga. Kao arhitekt kampanje, DM stvara pustolovine postavljajući pozicije čudovišta, zamka i blaga koje likovi drugih igrača mogu otkriti. Kao pripovjedač, DM pomaže ostalim igračima vizualizirati što se događa oko njih, improvizirajući kada pustolovi učine nešto ili odu negdje neočekivano. Kao glumac, DM igra uloge čudovišta i sporednih likova, time im daje život. A kao sudac, DM tumači pravila i odlučuje kada ih se treba pridržavati, a kada ih promijeniti. Svaki DM pristupa ulogama izmišljanja, pisanju, pripovijedanju, improvizaciji, glumi i suđenju drugačije i vjerojatno će igrači i sam DM u nekim uživati više nego u drugima.

D&D pravila pomažu DM-u i ostalim igračima da se dobro zabave, ali pravila nisu glavni autoritet. DM je glavni i upravlja igrom te smije "kršiti" ili mijenjati definirana pravila. Ipak, cilj DM-a nije pobiti pustolove, već stvoriti svijet kampanje koji se vrti oko njihovih djela i odluka, te osigurati da se igrači vraćaju po još.

Uspjev D&D sesije ovisi o sposobnosti DM-a da zabavi ostale igrače za stolom. Dok je njihova uloga stvaranje likova, udahnuti im život i pomoći usmjeravati kampanju kroz postupke svojih likova. DM treba održati igrače (i sebe) zainteresiranima i uživljenima u svijet koji je stvorio te omogućiti njihovim likovima da čine nevjerojatne stvari. Poznavanje u kojim dijelovima D&D-a igrači najviše uživaju pomaže DM-u stvoriti i voditi avanture u kojima će uživati i kojih će se sjećati. [6]

Aktivnosti na koje se DM može fokusirati za bolje zadovoljstvo igrača [6]:

- Gluma; igrači koji uživaju u glumi vole ulaziti u ulogu svog lika i govoriti njihovim klasom. Kao roleplayeri u duši, oni uživaju u društvenim interakcijama s NPC-ovima, čudovištima i ostalim članovima družine.
- Istraživanje; igrači koji žude za istraživanjem žele iskusiti čuda koja nudi fantastični svijet. Žele znati što se nalazi iza sljedećeg ugla ili brda, također vole pronalaziti skrivene tragove i blago

- Poticanje akcije; neki igrači vole poticati akciju, željni su da se stvari događaju čak i ako to znači preuzimanje opasnih rizika. Rađe će srljati u opasnost i suočiti se s posljedicama nego se suočiti s dosadom.
- Borba; drugi igrači uživaju u fantastičnoj borbi poput premlaćivanja zlikovaca i čudovišta. Traže bilo kakav izgovor da započne borba, preferirajući akciju nad pažljivim promišljanjem.
- Optimizacija lika; igrači koji uživaju u optimizaciji sposobnosti svojih likova vole podešavati svoje likove za vrhunske borbene performanse stjecanjem razina, novih značajki i čarobnih predmeta, rado će prihvatiti svaku priliku da pokažu nadmoć svojih likova
- Rješavanje problema; igrači koji žele rješavati probleme vole proučavati motivacije drugih likova, razmrsiti spletke zlikovaca, rješavati zagonetke i smišljati planove
- Pripovijedanje; igrači koji vole pripovijedanje žele doprinijeti priči. Sviđa im se kada su njihovi likovi uključeni u priču koja se razvija i uživaju u susretima koji su vezani s glavnom radnjom i proširuju je.

3.3. Sustav D20

Dvadeseto strana kockica je najčešće korištena za određivanje uspjeha ili neuspjeha radnji u D&D-u i tu kockicu nazivom d20. Ponekad se koriste dvanaesto strane, osmo strane, šesto strane, četvero strane kockice i novčić, one također imaju nazive ovisno o broju strana, d12, d10, d8, d6, d4 i d2.

Svi likovi i čudovišta u igri definirani su kroz šest osnovnih sposobnosti: snaga (engl. *strength*), spretnost (engl. *dexterity*), izdržljivost (engl. *constitution*), inteligencija (engl. *intelligence*), mudrost (engl. *wisdom*) i karizma (engl. *charisma*). Kod većine pustolova vrijednosti tih sposobnosti kreću se od 3 do 18, dok kod čudovišta mogu biti niske poput 1 ili visoke do 30. Te vrijednosti, kao i modifikatori koji iz njih proizlaze, temelj su gotovo svakoh bacanja d20 kockice

Tri glavne vrste bacanja čine srž pravila igre: provjera sposobnosti, bacanje za napad i bacanje za spas (engl. *saving throw*). Sve tri radnje slijede nekoliko koraka:

1. bacanje kockice i dodavanje modifikatora. Na rezultat bačene d20 kockice dodajemo odgovarajući modifikator. Najčešće je to modifikator jedne od šest sposobnosti, a ponekad uključuje i bonus stručnosti koji odražava specifičnu vještinu lika.
2. uračunavanje dodatnih bonusa i kazni. Značajke klase, čarolije, specifične okolnosti ili neki drugi efekti mogu dodati bonus na rezultat ili ga umanjiti.
3. usporedba rezultata s ciljanim brojem. Ako je ukupan zbroj jednak ili veći od ciljnog broj, radnja (provjera, napad, spašavanje) je uspješno, u suprotnom, nije uspjela.

DM obično određuje ciljne brojeve i govori igračima jesi li uspjeli odraditi radnju. DM ne treba otkriti igračima ciljani broj za uspjeh.

Ciljni broj za provjere sposobnosti i spas naziva se težina zadatka ili difficulty class (DC), dok se ciljni broj za napad naziva klasa oklopa ili Armour Class (AC). Ovo jednostavno pravilo temelj je rješavanja većine situacija u D&D-u.

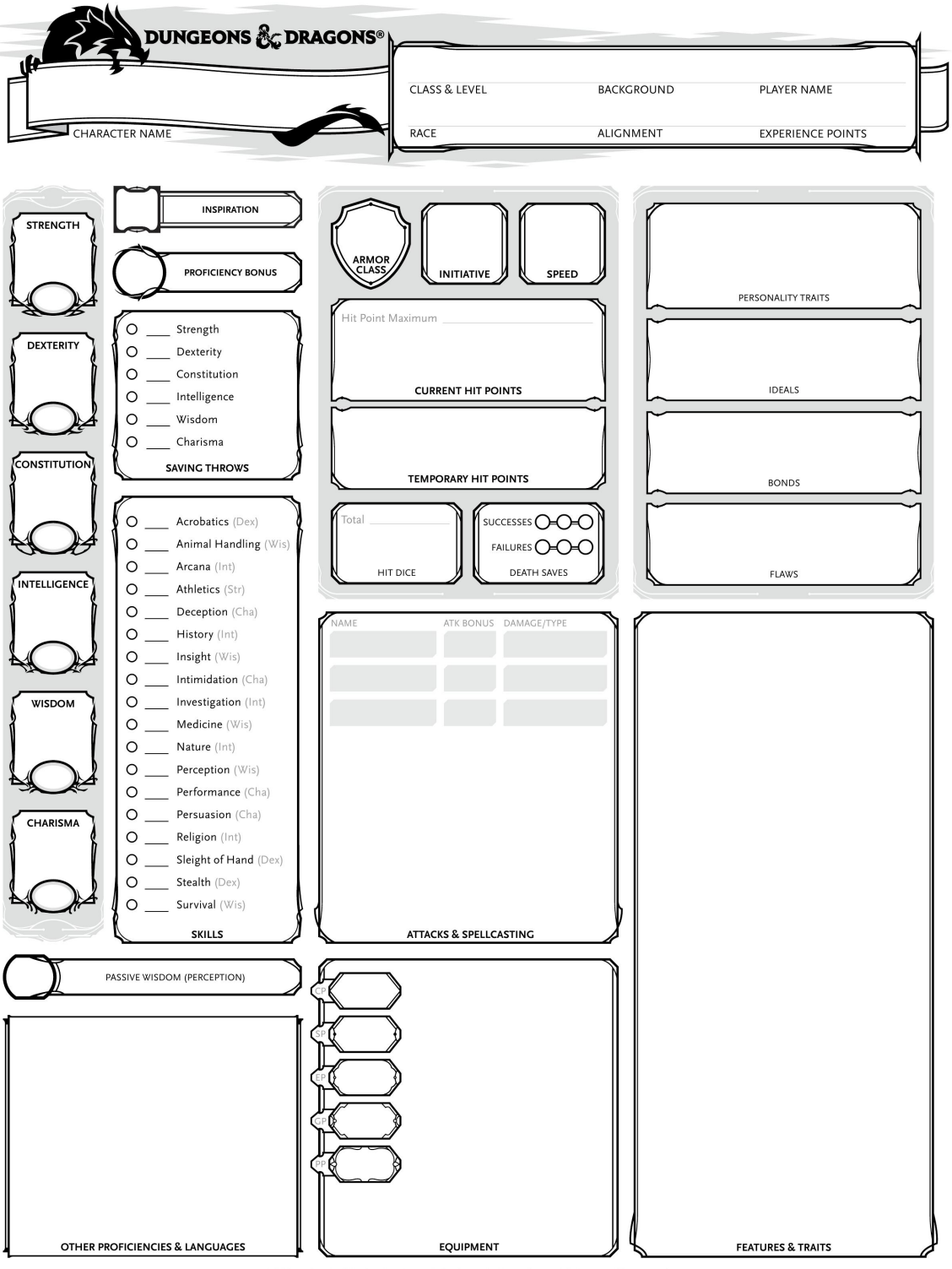
Kod nekih provjera sposobnosti, bacanja za napad ili bacanja za spas bacanje kockice se modificira posebnom situacijom koju nazivamo prednost (engl. *advantage*) ili odsutnost prednosti (engl. *disadvantage*). U ovoj situaciji kod bacanja D20 kockice, baca se dodatna D20 i ovisno o situaciji, ako je prednost ili odsutnost prednosti odabire se kockica s najvećom vrijednošću ili kockica s najmanjom vrijednošću.

3.4. Kreacija lika

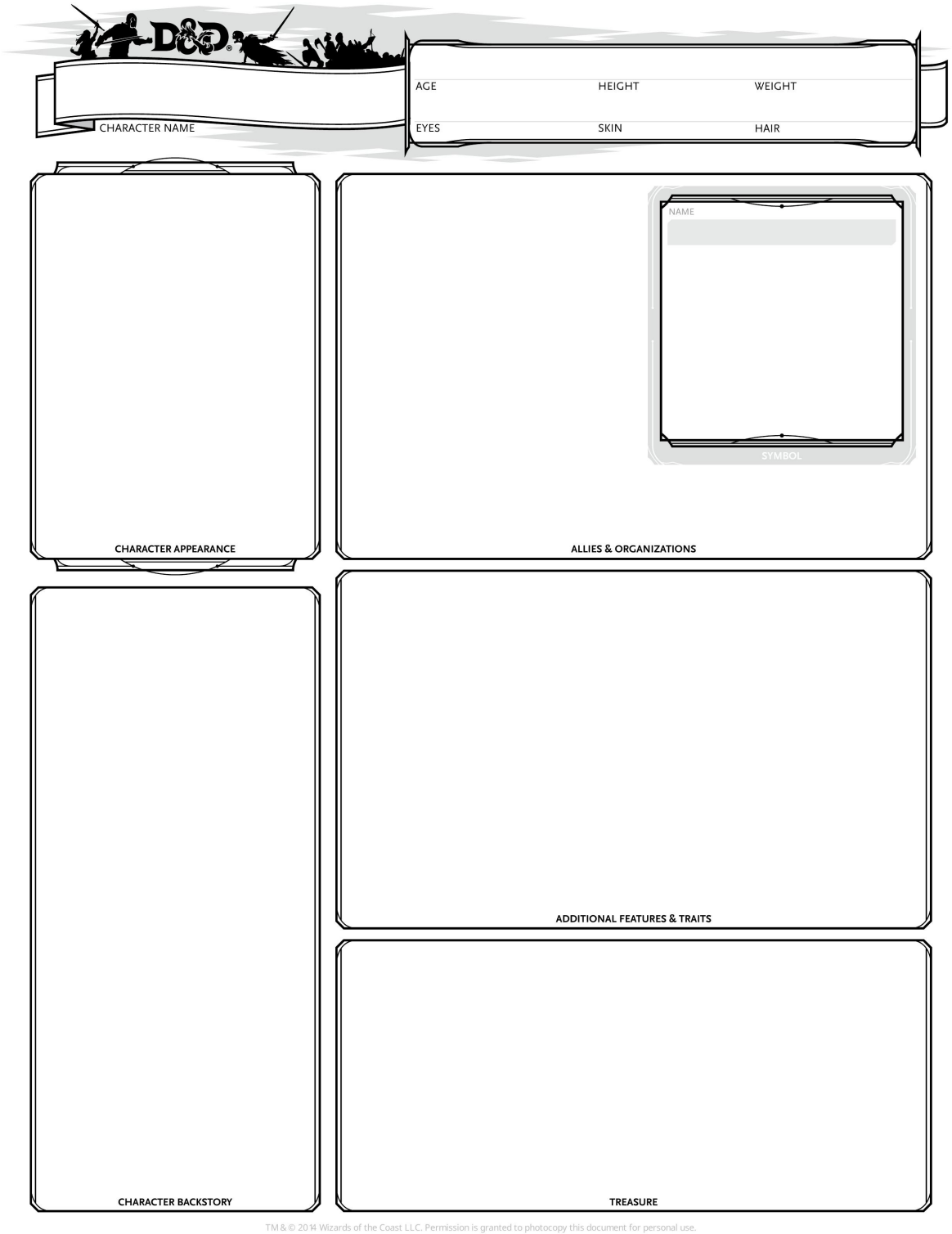
U ovom poglavlju će biti prikazana kreacija lika, odabir rase i klasa i ukratko prikazati pojedinosti koje likovi dobivaju odabirom rase i klase, te postupak kojim se dobivaju početne sposobnosti.

Na slici ispod se nalazi primjer prazne stranice lista o liku (engl. *character sheet*). Character sheet koristi igračima za zapisivanje i praćenje karakteristika njihovih likova. Kako ćemo prolaziti kroz različite dijelove kreacije lika odgovarajući podaci na listu će biti popunjeni, da bi kroz primjer prošli kroz kreaciju lika.

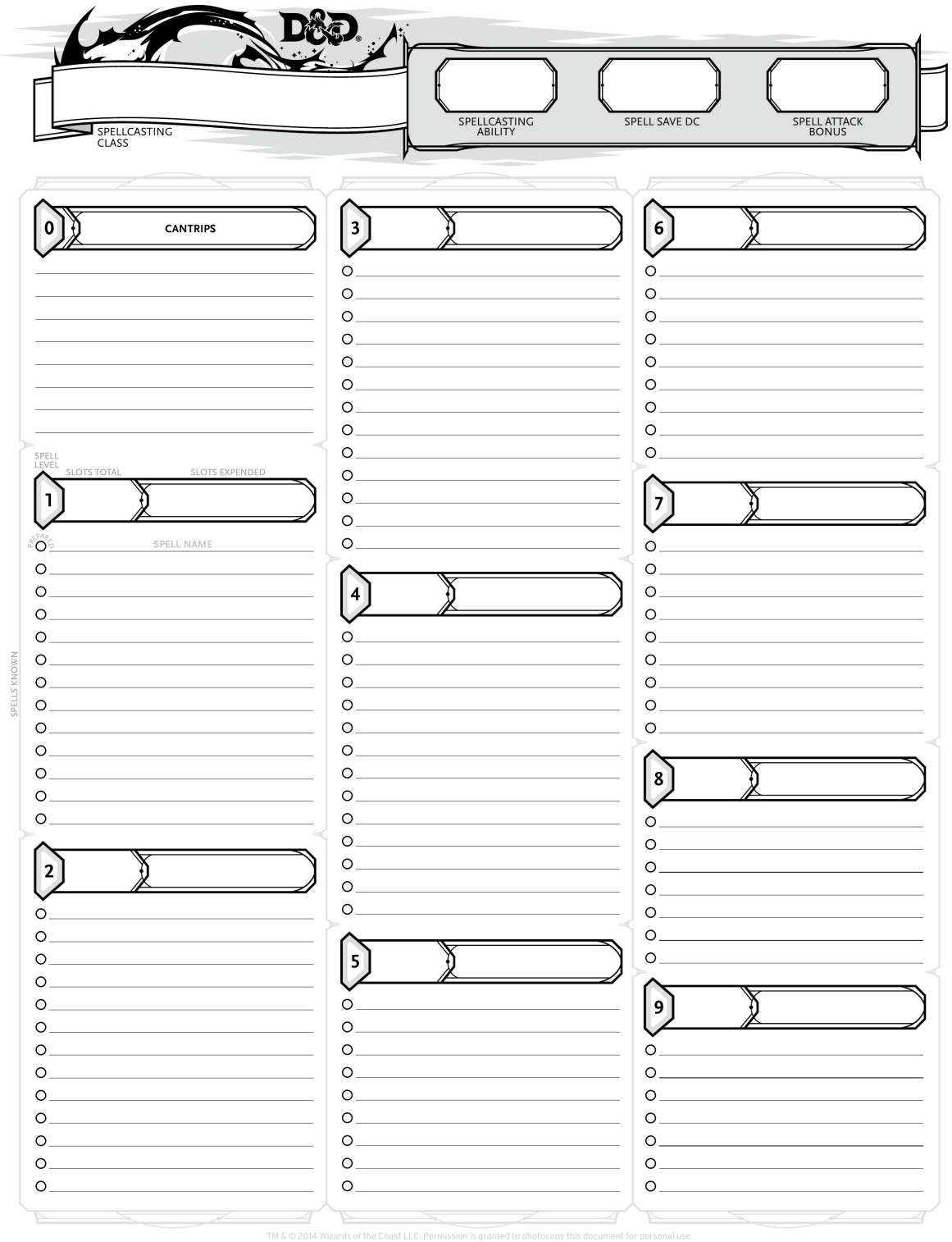
Character sheet se sastoji od tri stranice, prva stranica (Slika 1) sadrži podatke o sposobnostima lika, njegovu klasu, rasu, vještine, trenutnu opremu, životne bodove. Druga stranica (Slika 2) ima podatke o izgledu lika, starosti, visini, pozadinskoj priči, blagu. Omogućuje igraču da detaljno opiše svog lika, da se može lakše preuzeti ulogu lika te gledati fantastični svijet kroz oči lika. Dok treća stranica (Slika 2) sadrži podatke o čarolijama pripremljenim i znanim čarolijama, te razinama (engl. *levels*) i broj čarolija koje like može baciti prije potrebnog odmora.



Slika 1: Character sheet - Prva stranica [7]



Slika 2: Character sheet - Druga stranica [7]



Slika 3: Character sheet - Treća stranica [7]

3.4.1. Rase

Svaki lik pripada rasi, jednoj od mnogih rasa dostupnih u D&D-u. Najčešće rase likova su patuljci, vilenjaci i ljudi. Neke rase također imaju podrase, kao što su planinski patuljci ili

šumski vilenjaci. Odabrana rasa doprinosi identitetu lika, uspostavlja opći izgled, pruža prirodne talente stečene kulturom i precima. Rasa daje određene rasne osobine, kao što su posebna osjetila, vještine s određenim oružjem ili alatima, stručnost u jednoj ili više vještina ili sposobnost korištenja manjih čarolija. Ove osobine se ponekad podudaraju i nadopunjuju sposobnosti određenih klasa. Najčešće rasne osobine su uvećanje jedne ili više sposobnosti, dob lika, moralni kompas, veličina, brzina, jezici koje lik razumije i kojima se može služiti i razgovarati, te podrase; Članovi podrase imaju osobine matične rase uz dodatne osobine definirane podrasom.

Ovdje će biti definirane različite rase, njihove podrase, dob, veličina, brzina i ostale sposobnosti koje se dobivaju odabirom pojedine rase. Kao primjer će za patuljke biti navedeni detaljni podaci o rasi, dok će druge rase imati skraćenu podatke, razumijevanje svih različitih rasnih osobina nije nužno za implementaciju ni za razumijevanje igre:

- Patuljak (engl. *dwarf*) [5]
  - Povećanje sposobnosti: izdržljivost se povećava za 2.
  - Dob: patuljci sazrijevaju istom brzinom kao i ljudi, ali se smatraju mladima do 50. godine. Žive oko 350 godina.
  - Veličina: Srednja, patuljci su visoki između 4 i 5 stopa.
  - Brzina: brzina hoda je 25 stopa i ne smanjuje se nošenjem teškog oklopa
  - Dodatna svojstva rase:
    - Vid u mraku (engl. *darkvision*): patuljci vide u mraku i svijetlostno prigušenim uvjetima do 60 stopa.
    - Patuljačka otpornost (engl. *dwarven resilience*): prednost pri bacanju spasa protiv otrova i otpornost na otrov.
    - Patuljačka borbena obuka (engl. *dwarven combat training*): stručnost u korištenju bojne sjekire (engl. *battleaxe*), sjekiricom (engl. *handaxe*), lakim čekićem (engl. *throwing hammer*) i ratnim čekićem (engl. *warhammer*).
    - Stručnost s alatom (engl. *tool proficiency*): stručnost u korištenju jednog od vrsta alata: kovački alat, pivarski pribor, zidarski alat
    - Poznavanje kamena (engl. *stonecunning*): na sve provjere inteligencije (vezane uz vještinu povijest), dodaje se dvostruki bonus stručnosti na provjeru.
  - Jezici: patuljci govore, pišu i čitaju zajednički (engl. *common*) i patuljački
  - Podrase patuljaka: brdski patuljak; povećanje mudrosti za 1 i maksimalni broj životnih bodova povećava se za 1 po razini. Planinski patuljak; povećanje snage za 2 i stručnost s lakim i srednjim oklopom.

- Vilenjak (engl. *elf*)
  - Povećanje sposobnosti: spretnost se povećava za 2.
  - Dob: vilenjaci sazrijevaju istom brzinom kao i ljudi, ali se smatraju odraslima tek oko 100. godine. Žive do 750 godina.
  - Veličina: Srednja, vilenjaci su visoki od ispod 5 do preko 6 stopa.
  - Brzina: brzina hoda je 30 stopa.
  - Tri podrase: visoki vilenjak (engl. *high elf*); Šumski vilenjak (engl. *wood elf*); Tamni vilenjak (engl. *drow*);
- Halfling
  - Povećanje sposobnosti: spretnost se povećava za 2.
  - Dob: halflinzi dosežu odraslu dob s 20 godina i žive do oko 150 godina.
  - Veličina: Mala, halflinzi su visoki oko 3 stope.
  - Brzina: brzina hoda je 25 stopa.
  - Jezici: govore, pišu i čitaju zajednički i halflingški (engl. *halfling*).
- Čovjek (engl. *human*)
  - Povećanje sposobnosti: sve sposobnosti se povećavaju za 1.
  - Dob: ljudi dosežu odraslu dob u kasnim tinejdžerskim godinama i žive manje od stoljeća.
  - Veličina: Srednja, ljudi znatno variraju u visini i građi.
  - Brzina: brzina hoda je 30 stopa.
  - Jezici: ljudi govore, pišu i čitaju zajednički i jedan dodatni jezik po izboru.
- Dragonborn
  - Povećanje sposobnosti: snaga se povećava za 2, a karizma za 1.
  - Dob: brzo rastu, odrasli su s 15 godina i žive oko 80 godina.
  - Veličina: Srednja, viši su i teži od ljudi, često preko 6 stopa.
  - Brzina: brzina hoda je 30 stopa.
  - Jezici: govore, pišu i čitaju zajednički i zmajski (engl. *draconic*).
- Gnom (engl. *gnome*)
  - Povećanje sposobnosti: inteligencija se povećava za 2.
  - Dob: gnomovi odrastaju oko 40. godine i žive 350 do 500 godina.
  - Veličina: Mala, visoki su između 3 i 4 stope.
  - Brzina: brzina hoda je 25 stopa.

  - Jezici: gnomovi govore, pišu i čitaju zajednički i gnomski (engl. *gnomish*).
  - Podvrste: šumski gnom (engl. *forest gnome*) Kameni gnom (engl. *rock gnome*).

- Poluvilenjak (engl. *half-elf*)
  - Povećanje sposobnosti: karizma se povećava za 2 i dvije druge sposobnosti po izboru za 1.
  - Dob: sazrijevaju kao ljudi, odrasli s 20 godina, žive preko 180 godina.
  - Veličina: Srednja, otprilike iste veličine kao ljudi.
  - Brzina: brzina hoda je 30 stopa.
  - Jezici: poluvilenjaci govore, pišu i čitaju zajednički, vilinski (engl. *elvish*) i jedan dodatni jezik po izboru.

- Poluork (engl. *half-orc*)
  - Povećanje sposobnosti: snaga se povećava za 2, a izdržljivost za 1.
  - Dob: sazrijevaju brže od ljudi s oko 14 godina, žive do 75 godina.
  - Veličina: Srednja, krupniji su od ljudi.
  - Brzina: brzina hoda je 30 stopa.
  - Jezici: poluorkovi govore, pišu i čitaju zajednički i orkovski (engl. *orc*).

- Tiefling
  - Povećanje sposobnosti: inteligencija se povećava za 1, a karizma za 2.
  - Dob: sazrijevaju kao ljudi, žive malo duže.
  - Veličina: Srednja, iste veličine i građe kao ljudi.
  - Brzina: brzina hoda je 30 stopa.
  - Jezici: tieflinzi govore, pišu i čitaju zajednički i pakleni (engl. *infernal*).

3.4.2. Klase

Klasa je temelj onoga što likovi mogu učiniti. Ona je više od pukog zanimanja; ona je životni poziv lika. Klasa lika oblikuje način na koji igrači doživljavaju svijet i kako s njim komuniciraju. Klasa daje raznoliko posebne značajke, poput borčeve vještine s oružjem i oklopom ili magovim čarolijama. Na nižim razinama, klasa nudi tek dvije ili tri značajke, no kako lik napreduje, stječe nove, a postojeće se često poboljšavaju [5]. U tablici ispod se nalaze nazivi klasa, opisi, kockice za životne bodove koje svaka klasa koristi kod određivanja životnih bodova po razini, primarne sposobnosti klase, sposobnosti koje se koriste kod bacanja za spas, te stručnosti s oklopom i oružjem. Osim ovih sposobnosti svaka klasa ima specifične značajke no kroz njih nećemo prolaziti u ovom radu i dostupne su u `Player's Handbook`-u [5].

Tablica 1: Tablica klasa

Klasa	**Opis**	**Vrsta kockica za životne bodove**	**Primarna sposobnost**	**Bacanja za spas**	**Stručnosti s oklopom i oružjem**
Barbarin	Žestoki ratnik primitivnog podrijetla koji može ući u bojni bijes	d12	Snaga	Snaga, izdržljivost	Laki i srednji oklopi, štitovi, jednostavno i borbeno oružje
Bard	Nadahnjujući mađioničar čija moć odjekuje glazbom stvaranja	d8	Karizma	Spretnost, karizma	Laki oklop, jednostavno oružje, ručni samostreli, dugi mačevi, rapiri, kratki mačevi
Klerik	Svećenički prvak koji upravlja božanskom magijom u službi više sile	d8	Mudrost	Mudrost, karizma	Laki i srednji oklopi, štitovi, jednostavno oružje
Druid	Svećenik "Stare Vjere", upravlja moćima prirode i poprima životinjske oblike	d8	Mudrost	Inteligencija, mudrost	Laki i srednji oklopi (nemetalni), štitovi (nemetalni), toljage, bodeži, strelice, koplja, mlatovi, štapovi, sablje, srpovi, praćke
Borac	Majstor borbe, vješt s raznim oružjem i oklopima	d10	Snaga ili spretnost	Snaga, izdržljivost	Svi oklopi, štitovi, jednostavno i borbeno oružje
Redovnik	Majstor borilačkih vještina koji koristi snagu tijela za fizičko i duhovno savršenstvo	d8	Spretnost, mudrost	Snaga, spretnost	Jednostavno oružje, kratki mačevi
Paladin	Sveti ratnik vezan svetom zakletvom	d10	Snaga, karizma	Mudrost, karizma	Svi oklopi, štitovi, jednostavno i borbeno oružje
Rendžer	Ratnik koji koristi borilačku vještinu i magiju prirode za borbu protiv prijetnji	d10	Spretnost, mudrost	Snaga, spretnost	Laki i srednji oklopi, štitovi, jednostavno i borbeno oružje
Lupež	Nitkov koji koristi skrivanje i trikove kako bi svladao prepreke i neprijatelje	d8	Spretnost	Spretnost, inteligencija	Laki oklop, jednostavno oružje, ručni samostreli, dugi mačevi, rapiri, kratki mačevi
Čarobnjak	Bacač čarolija koji crpi urođenu magiju iz dara ili krvne loze	d6	Karizma	Izdržljivost, karizma	Bodeži, strelice, praćke, štapovi, laki samostreli
Vještac	Korisnik magije koja proizlazi iz pogodbe s izvanplanarnim entitetom	d8	Karizma	Mudrost, karizma	Laki oklop, jednostavno oružje
Mag	Učeni korisnik magije sposoban manipulirati strukturama stvarnosti	d6	Inteligencija	Inteligencija, mudrost	Bodeži, strelice, praćke, štapovi, laki samostreli

(Izvor: Player's Handbook [5])

3.4.3. Ostale značajke

U ostale značajke lika ubrajamo ime, spol, visinu i težinu, opredjeljenje, jezike, osobne karakteristike (osobine, ideale, veze, mane) i pozadinu. Značajke opisuju način na koji bi se lik mogao ponašati tijekom igre. Značajke kao ime, spol, visinu, težinu i osobne karakteristike igrač sam kreira kako bi pobliže opisao svog lika i njegovo ponašanje u svijetu, dok se značajke kao što su opredjeljenje, pozadina mogu birati (ili igrači/DM mogu kreirati pozadine), te će u nastavku ukratko biti opisani.

Opredjeljenje u opisuje likove moralne i osobne stavove, te je kombinacija dvaju faktora: jedan identificira moralnost (dobar, zao, neutralan), a drugi opisuje odnos prema društvu i redu (zakonit, kaotičan, neutralan). Svaka kombinacija ovih dva faktora ukupno definira devet mogućih kombinacija. Ovi devet opredjeljenja opisuju tipično ponašanje s tim opredjeljenjem, pojedinci mogu značajno odstupati od tog tipičnog ponašanja [5]:

- Zakonito dobar (engl. *Lawful good*) - pojedinci će učiniti ispravnu stvar onako kako to društvo očekuje
- Neutralno dobar (engl. *Neutral good*) - čine najbolje što mogu kako bi pomogli drugima u skaldu s svojim potrebama
- Kaotično dobar (engl. *Chaotic good*) - djeluju kako im savjest nalaže, s malo obzira prema onome što drugi očekuju
- Zakonito neutralan (engl. *Lawful neutral*) - pojedinci djeluju u skaldu sa zakonom, tradicijom ili osobnim kodeksima
- Neutralan (engl. *Neutral*) - opredjeljenje onih koji radije izbjegavaju moralna pitanja i ne zauzimaju strane, čineći ono što se u tom trenutku čini najboljim
- Kaotično neutralan (engl. *Chaotic neutral*) - slijede svoju volju, držeći svoju osobnu slobodu iznad svega
- Zakonito zao (engl. *Lawful evil*) - uzimaju ono što žele, unutar granica kodeksa tradicije, lojalnosti ili reda
- Neutralno zao (engl. *Neutral evil*) - opredjeljenje onih koji čine sve što im može proći nekažnjeno, bez suosjećanja ili grižnje savjesti
- Kaotično zao (engl. *Chaotic evil*) - djeluju s proizvoljnim nasiljem, potaknuti pohlepom, mržnjom ili željom za krvlju

Svaka priča ima neki početak, te pozadina lika otkriva odakle lik dolazi, kako su postali pustolov i koje je njihovo mjesto u svijetu. Odabir pozadine pruža važne smjernice o identitetu lika, te daje liku dodatne značajke kao stručnost u nekoj vještini ili znanje nekog jezika. U samom primjeru izrade lika će biti detaljno opisana jedna pozadina, te sve dodatne značajke koje lik dobiva i opremu s kojom počne. Kratki popis pozadina [5]:

- Akolit (engl. *Acolyte*) - osoba koja je provela život u službi hrama određenog boga, djelujući kao posrednik između svetog i smrtnog svijeta
- Šarlatan (engl. *Charlatan*) - osoba koja zna manipulirati ljudima i koristi trikove, prevare i lažne identitete za vlastitu korist
- Kriminalac (engl. *Criminal*) - iskusni prijestupnik s poviješću kršenja zakona i s kontaktima u podzemlju
- Zabavljač (engl. *Entertainer*) - osoba koja živi za nastup pred publikom i može ih zabaviti, očarati ili inspirirati
- Narodni heroj (engl. *Folk hero*) - osoba iz skromnog društvenog položaja, kojeg ljudi smatraju svojim prvakom protiv tiranije i čudovišta
- Pustinjak (engl. *Hermit*) - osoba koja živi u samoći, pronalazeći tišinu i odgovore daleko od društva
- Plemić (engl. *Noble*) - osoba koja nosi plemićku titulu, posjeduje zemlju ili ima politički utjecaj i razumije bogatstvo i moć
- Mudrac (engl. *Sage*) - osoba koja je provela godine učeći o znanju svijeta i proučavajući rukopise
- Mornar (engl. *Sailor*) - osoba koja je godinama plovila na morskom plovilu, suočavajući se s olujama i čudovišta
- Vojnik (engl. *Soldier*) - osoba kojoj je rat bio život, obučena u borbi i preživljavanju na bojnom polju
- Uličar (engl. *Urchin*) - osoba koja je odrasla sama na ulici, siromašna i bez roditelja, preživljavajući zahvaljujući svojoj snalažljivosti

3.5. Primjer lika

U ovom poglavlju ćemo proći kroz kreaciju lika po pravilima definiranim u prijašnjim poglavljima. Lik će početi na prvoj razini.

Kreacija počinje s odabirom rase. Za rasu odabiremo poluorka, što znači da će lik imati povećanu snagu i izdržljivost, snagu za dva boda, izdržljivost za jedan, brzinu od 30 stopa, te dobiva osobine vid u mraku (engl. *darkvision*) i prijeteću prisutnost (engl. *menacing*). Prijeteća prisutnost omogućava stručnost u vještini zastrašivanja/prijetnje (engl. *intimidation*). Još dodatne dvije sposobnosti koje daje poluork rasa su nemilosrdna izdržljivost (engl. *relentless endurance*); kad je liku broj životnih bodova smanjen na nulu, nije odmah ubijen, već pada na jedan životni bod i ne može opet koristiti ovu sposobnost dok ne završi dugi odmor. Druga je divljački napadi (engl. *savage attacks*); kad lik postigne kritični pogodak napadom oružja u bliskoj borbi, može bacati još jednu dodatnu kockicu od kockica štete oružja i dodati vrijednost šteti kritičnog pogotka.

Za klasu odabiremo barbarina, što znači da će lik koristiti kockicu D12 za akcije vezane uz njegove životne bodove, npr. kod prelaska na višu razinu, ili kod kratkog/dužeg odmora. Lik dobiva stručnosti s lakim i srednjim oklopima, štitovima, jednostavnim i borbenim oružjima. Kod bacanja spasa koristi snagu i izdržljivost, te odabira stručnosti u dvije vještine iz područja rukovanja životinjama (engl. *animal handling*), atletika (engl. *athletics*), zastrašivanja, prirode (engl. *nature*), percepcije (engl. *perception*) i preživljavanja (engl. *survival*). Za ovog lika odabiremo stručnost u vještinama rukovanja životinjama i percepcije, te je lik također vješt u zastrašivanju zbog odabrane rase. Jedinstvena sposobnost barbarina je bijes. Tokom bitke barbarin može pobjesniti korištenjem bonus akcije. Dok bjesni dobiva dodatne pogodnosti, ako ne nosi teški oklop; ima prednost na provjere snage i bacanja za spašavanje snage; kad napada oružjem za blisku borbu, koje koristi snagu, dobiva bonu na bacanje štete; ima otpornost na neke vrste štete, otpornost na tučenje (engl. *bludgeoning*), na probijanje (engl. *piercing*) i na sječenje (engl. *slashing*). Zadnja sposobnost koju lik dobiva od klase s prvom razinom je neoklopljena obrana (engl. *armored defense*); dok lik ne nosi oklop, klasa oklopa (AC) je jednak 10 + modifikator spretnosti + modifikator izdržljivosti [5]. Na višim razinama klase lik dobiva još više mogućnosti no ovdje nećemo proći kroz njih.

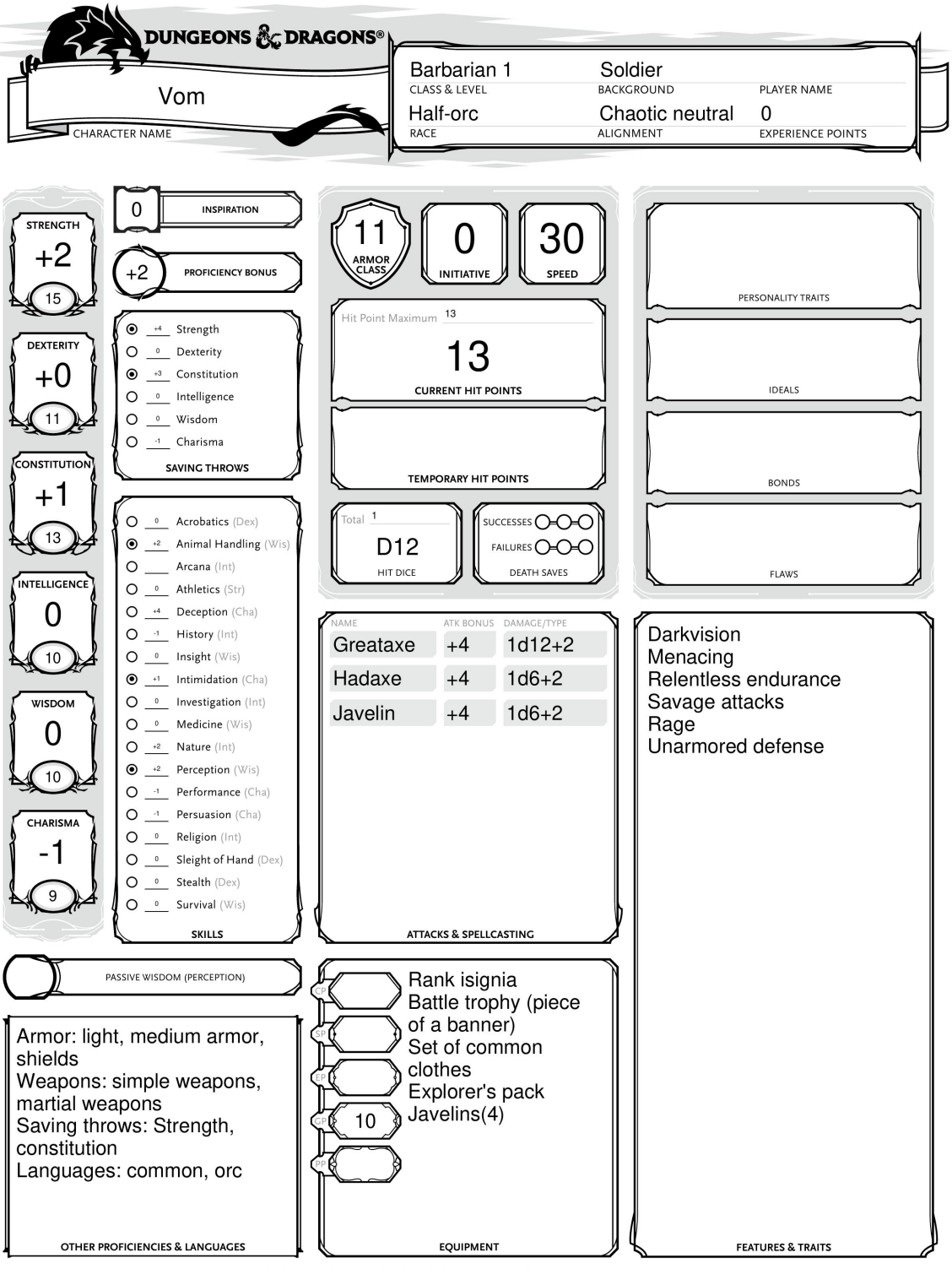
Slijedeće je potrebno odrediti vrijednost i modifikator svake sposobnosti i maksimum životnih bodova. Postoji više načina na koji se mogu odrediti početne vrijednosti za sposobnosti, no u *Player's Handbook-u* je preporučeno da se šest puta bace četiri D6 kockice, te se kod svakog bacanja izbaci najmanja kockica. Zbroj triju kockica određuje vrijednost jedne od sposobnosti, te nakon šest bacanja igrač može proizvoljno rasporediti vrijednosti po sposobnostima. Modifikatori za sposobnosti se isčitavaju iz slijedeće tablice [5]:

Tablica 2: Vrijednosti sposobnosti i modifikatori

Vrijednost	**Modifikator**	**Vrijednost**	**Modifikator**
1	-5	16-17	+3
2-3	-4	18-19	+4
4-5	-3	20-21	+5
6-7	-2	22-23	+6
8-9	-1	24-25	+7
10-11	0	26-27	+8
12-13	+1	28-29	+9
14-15	+2	30	+10

(Izvor: Player's Handbook [5])

Maksimalni broj životnih bodova na prvoj razini određuje vrsta klase, pošto je ovaj lik barbarin, broj životnih bodova će iznositi 12 + modifikator izdržljivosti. Ispod se nalazi slika s primjerom popunjenog character sheet-a za ovog lika, neki dijelovi su popunjeni, a nisu ovdje opisani, pošto su manje važni za razumijevanje igre i direktno se isčitavaju iz pravila (npr. oprema koju lik dobiva).



Slika 4: Primjer popunjenog character sheet-a

3.6. Borba

Ovo poglavlje pruža pravila koja su potrebna kako bi likovi i čudovišta sudjelovali u borbi, bilo to da se radi o kratkom okršaju ili produženom sukobu. Kroz cijelo poglavlje pravila se odnose na igrača ili DM-a. DM kontrolira sva čudovišta i likove koji nisu igrači, a uključeni su u borbu, drugi igrači kontroliraju svoje likove. Primjer borbe će biti prikazan u poglavlju 5.4.

Tipičan sukob je između dvije strane i organiziran je u ciklus rundi i poteza. Runda predstavlja oko šest sekundi u svijetu igre. Tijekom runde, svaki sudionik u bitci ima svoj potez. Redoslijed poteza određuje se na početku borbenog susreta, kada svi bacaju inicijativu. Nakon što svi odrade potez, borba se nastavlja u slijedeću rundu ako nijedna strana nije pobijedila drugu.

Prije početka borbe DM određuje tko bi mogao biti iznenađen na početku borbe. Ako nijedna strana ne pokušava biti prikrivena automatski primjećuju jedna drugu i ovaj korak se preskače, no ako se jedna strana uspješno prikrade ostvaruje prednost. Strana koje je iznenađena, tj. lik ili čudovište koje je iznenađeno se ne može pomaknuti ili poduzeti akciju u svom prvom potezu borbe, također ne može poduzeti reakciju dok taj prvi potez ne završi. Član grupe može biti iznenađen čak i ako ostali članovi nisu [5].

Inicijativa određuje redoslijed poteza tokom borbe. Kada borba počne svaki sudionik radi provjeru spretnosti (engl. *dexterity check*) kako bi odredio svoje mjesto u poretku inicijative. Baca D20 kockicu i na dobivenu vrijednost dodaje modifikator spretnosti. DM određuje inicijativu za sve likove, čudovišta koje ne kontroliraju drugi igrači. Nakon dobivenih vrijednosti za inicijativu, DM ih rangira od najviše do najmanje vrijednosti te određuje redoslijed poteza. U slučajevima kad igrači imaju istu inicijativu mogu međusobno odrediti tko nastupa prvu, ako čudovište i igrač imaju jednaku inicijativu DM određuje redoslijed.

3.6.1. Igračev potez

Kad je igrač na potezu može pomicati svog lika do udaljenosti određenom brzinom lika i odraditi jednu akciju. U pravilu svako polje je veličine pet sa pet stopa, što bi značilo da se lik sa brzinom od trideset stopa može pomaknuti maksimalno šest polja. Igrač odlučuje ako će se prvo kretati ili odraditi akciju. U poglavlju 3.6.3 će biti opisane najčešće akcije koje likovi mogu odraditi, mnoge značajke različitih klasa i druge sposobnosti pružaju dodatne opcije za akcije.

Razne značajke klase, čarolije i druge sposobnosti omogućavaju poduzimanje dodatne akcije u trenutnom potezu. Dodatna akcije se može jedino poduzeti u slučaju kad značajka klase, čarolija ili neka druga sposobnost navodi da se nešto može odraditi kao dodatna akcija. Po potezu, jednako kao s akcijom, može se odraditi samo jedna dodatna akcija, te ako lik ima dostupno više dodatnih akcija potrebno je odrediti koju je najbolje iskoristiti u tom slučaju [5, str. 189].

Kao dio poteza lik može stupiti u interakciju s jednim predmetom ili značajkom okoline, ta interakcija se može dogoditi tijekom kretanja ili akcije. Na primjer, otvaranje vrata dok se lik kreće prema neprijatelju ili izvlačenje oružja kao dio iste akcije za napad bi spadale pod

interakciju s okolinom.

Određene posebne sposobnosti, čarolije i situacije omogućuju poduzimanje posebne akcije zvane reakcija. Reakcija je trenutni odgovor na okidač neke vrste, koji se može dogoditi tokom poteza igrača ili tuđeg poteza. Napad prilike (engl. *Opportunity attack*) je najčešća vrsta reakcije i više je opisana u poglavlju 3.6.4.

3.6.2. Kretanje i položaj

Kad je lik na potezu može se pomaknuti do udaljenosti definirane brzinom lika. Po potezu lik može potrošiti proizvoljnu količinu brzine, ne mora potrošiti svu ili uopće iskoristiti brzinu. Kretanje se može razdijeliti kroz cijeli potez. Moguće je koristiti dio brzine prije i poslije akcije. Na primjer, ako je brzina lika trideset stopa, može se kretati deset stopa, iskoristiti neku akciju, a zatim se kretati još dvadeset stopa. Ako akcija uključuje više od jednog napada oružjem, kretanje se može dodatno razdvojiti između tih napada.

Borbe se rijetko odvijaju u praznim sobama. Špilje pune kamenja i guste šume su tipična mjesta odvijanja borbe i sadrže težak teren (engl. *difficult terrain*). Za svako kretanje kroz ovakav teren potrebno je utrošiti jednu dodatnu stopu za svaku stopu koju lik želi preći. Na primjer, ako se radi o udaljenosti od pet stopa, za prolaz kroz težak teren potrebno je utrošiti deset stopa brzine [5, str. 190].

Likovi se mogu kretati kroz prostor prijateljskih stvorenja, ali ne mogu završiti potez ili zaustaviti se na istom polju. Kroz prostor neprijateljskih stvorenja mogu se kretati samo ako je stvorenje barem dvije veličine veće ili manje od lika koji se kreće, no nije dozvoljeno završavanje kretanja u prostoru koji zauzima to stvorenje [5, str. 191].

U tablici ispod se nalaze veličine stvorenja i različite količine prostora koje zauzimaju, tokom borbe veća stvorenja kontroliraju veću količinu prostora, također objekti u okolini ponekad mogu koristiti kategorije veličine. Prostor stvorenja je područje u stopama koje to stvorenje kontrolira tokom borbe, a nije izraz njihovih fizičkih dimenzija.

Tablica 3: Kategorije veličine

Veličina	**Prostor**
Sićušan	2.5 sa 2.5 stopa
Mala	5 sa 5 stopa
Srednja	5 sa 5 stopa
Velika	10 sa 10 stop
Ogromna	15 sa 15 stopa
Gigantska	20 sa 20 stopa ili više

(Izvor: Player's Handbook [5, str. 191])

3.6.3. Akcije

Kad je lik na potezu i želi odraditi neku akciju, osim akcija koje dobiva od klase ili nekih drugih posebnih značajki uvijek može odraditi neku od slijedećih akcija ili improvizirati akciju. Ako je akcija improvizirana DM odlučuje ako je moguća, koju vještinu lik može iskoristiti za akciju i kakvo bacanje kockice će odrediti uspješnost akcije.

Napad je najčešća akcija u borbi, bilo da se zamahuje mačem, ispaljuje strijela iz luka ili tučnjava šakama, ovom akcijom izvodi se jedan napad izbliza ili napad na daljinu [5, str. 192]. Napadi će biti detaljnije opisani u poglavlju 3.6.4.

Bacanje čarolija, mnoge vrste čudovišta i likova imaju pristup čarolijama i mogu iv koristiti u borbi. Svaka čarolija ima definirano vrijeme bacanja (engl. *cast time*), koje određuje mora li bacač koristiti akciju, reakciju, minute ili sate vremena da baci čaroliju. U nekim slučajevima, kao kad čarolija ima vrijeme bacanja od više sati, bacanje čarolije nije nužno jedna akcija. Većina čarolija ima vrijeme bacanja od jedne akcije, pa je često takva čarolija korištena tokom borbe [5, str. 192].

Trk (engl. *dash*), omogućava liku da potroši akciju kako bi se mogao na istom potezu dodatno kretati. Povećanje je jednako brzini lika, nakon što se primjene bilo kakvi modifikatori brzine. Na primjer, s brzinom od trideset stopa, na istom potezu moguće je pomaknuti se šestdeset stopa ako se iskoristi ova akcija. Bilo kakve promjene na brzinu također imaju utjecaj na dodatnu brzinu dobivenu ovom akcijom, na primjer ako je lik usporen i ima smanjenu brzinu s trideset na petnaest, korištenjem ove akcije će dobiti dodatnih petnaest brzine [5, str. 192].

Isključivanje iz borbe (engl. *disengage*), akcija omogućuje kretanje bez da neprijatelji dobivaju napad prilike kad se lik kreće oko njih [5, str. 192].

Izmicanje (engl. *dodge*), akcija omogućava usredotočenost na izbjegavanje napada. Do početka slijedećeg poteza, svako bacanje napada protiv lika koji je iskoristio ovu akciju, ima odsutnost prednosti ako lik vidi napadača, a sva bacanja za spas na spretnost se izvode s prednošću [5, str. 192].

Pomoć, likovi mogu pružiti pomoć drugom liku u izvršavanju nekog zadatka. Korištenjem ove akcije drugi lik dobiva prednost na slijedeću provjeru sposobnosti koju izvrši za obavljanje zadatka u kojem dobiva pomoć, pod uvjetom da napravi tu radnju prije nego pomagač opet dođe na potez. Također moguće je pomoći prijateljskom stvorenju u napadu na neprijatelja unutar pet stopa, napadač na svom potezu dobiva prednost na prvo bacanje za napad [5, str. 192].

Pretraga, akcija može biti iskorištena za pretragu okoline i pronalaženju nečega. Ovisno o prirodi pretrage, DM može zatražiti da se napravi provjera vještine percepcija ili istraga [5, str. 193].

3.6.4. Napadi

Napade možemo podijeliti u nekoliko vrsta, napadi na blizinu, na daljinu i napadi prilike. Svaka vrsta napada se ima razlike no svi imaju jednaku strukturu i način na koji se izvode [5, str. 193-194]:

1. Odabir mete; potrebno je odabrati metu unutar dometa napada, meta može biti stvorenje, predmet ili lokacija. Na primjer, lokacija se može odabrati ako pokušavamo napasti nevidljivog neprijatelja ili ako čarolija koju bacamo ima utjecaj na veće područje.

2. Određivanje modifikatora; DM određuje ako se meta skriva iza zaklona i ako napadač ima prednost ili nema prednost protiv mete. Uz to, čarolije, posebne sposobnosti i drugi učinci mogu imati utjecaja.

3. Razriješivanje napada; igrač radi bacanje za napad (engl. *attack roll*), ovo bacanje određuje ako je meta uspješno pogođena napadom ili je napad promašio. U slučaju pogotka određuje se šteta koja je napravljena meti, također bacanjem kockice, ovisno o napadu koriste se različite kockica, neki napadi uzrokuju posebne učinke uz štetu ili umjesto nje.

Za izvođenje bacanja za napad koristi se d20 kockica te se na dobivenu vrijednost dodaju određeni modifikatori. Ako je ukupan zbroj bacanja i modifikatora jednak ili veći od klase oklopa mete, u tom slučaju napad pogađa. Dva najčešća modifikatora koji se dodaju su modifikator sposobnosti (engl. *ability modifier*) i bonus stručnosti lika. Modifikator sposobnosti odgovara sposobnosti koja se koristi za napad, npr. može biti snaga ili spretnost ovisno o vrsti oružja. Bonus stručnosti se dodaje ako lik ima stručnost u korištenju oružja kojim napada.

U rjeđim slučajevima kao rezultat bacanja d20 moguće je dobiti 1 ili 20. Ako je dobivena vrijednost 20, napad je uvijek pogodak, neovisno o bilo kakvim modifikatorima ili klasi oklopa mete, također ovakav napad je uvijek kritičan pogodak. Ako je dobivena vrijednost 1, napad uvijek promašuje i ne dodaje se modifikator ni ne provjerava klasa oklopa mete. Slična funkcija je kod ostalih bacanja d20, vrijednost od 20 je skoro uvijek uspjeh dok je vrijednost 1 neuspjeh.

3.6.5. Šteta i iscjeljivanje

Ozljede i rizik od smrti stalni su pratitelji onih koji istražuju svjetove D&D-a. Životni bodovi predstavljaju kombinaciju fizičke i mentalne izdržljivosti. Stvorenje s više životnih bodova teže je ubiti ili onesvijestiti. Ona s manje životnih bodova su krhkija. Trenutni životni bodovi stvorenju mogu biti bilo koji broj između nule i maksimuma životnih bodova. Kad kod stvorenje pretrpi štetu, ta šteta oduzima od životnih bodova, gubitak životnih bodova nema utjecaj na sposobnosti stvorenja sve dok je vrijednost veća od nula.

Ako šteta ne rezultira smrću stvorenja, tada nije trajna. Likovi mogu vratiti životne bodove odmorom, korištenjem čarolija ili ispijanjem ljekovitog napitka. Kada stvorenje primi iscje-

ljivanje bilo koje vrste, vraćeni životni bodovi dodaju se trenutnim životnim bodovima i ne mogu premašiti maksimum životnih bodova tog lika ili stvorenja.

Postoji nekoliko situacije koje se mogu dogoditi kad neki lik ili stvorenje padne na nula životnih bodova. Manje važna stvorenja, čudovišta uglavnom umiru odmah nakon što im životni bodovi padnu na nulu, dok važniji likovi mogu pasti u nesvijest ili imati nekoliko pokušaja da se spase od smrti i stabiliziraju svoje stanje. U slučaju kad količina štete svede lika na nula životnih bodova i ostatak štete je veći ili jednak broju maksimalnih životnih bodova lika, lik umire u tom trenutku. Padom u nesvijest tijekom borbe znači da lik ili stvorenje na svom potezu mora izvršiti bacanje za spas od smrti (engl. *death save*). Za spas se koristi d20 i sve vrijednosti jednake ili veće od deset približavaju lika životu, dok sve ispod deset smrti. Na treći uspjeh lik postaje stabilan, te se više ne mora spašavati od smrti, dok na treći neuspjeh lik umire. Bacanje dvadeset ili jedan se broji kao dva uspijeha ili neuspjeha. Stabilizirano stvorenje ima nula životnih bodova i onesviješteno je, odmorom se životni bodovi mogu povratiti, no odmor traje neko vrijeme i nije moguć tokom borbe. Tokom borbe iscjeljivanje na barem jedan životni bod je jedini način za osvijestiti lika ili stvorenje.

4. Prikaz implementacije

U ovom poglavlju će biti opisani alati i tehnologije korištene za implementaciju platforme. Korišteno je više različitih tehnologija, fokus je na odabir i korištenje tehnologija koje su što više prikladne za implementacije web aplikacija. Ovime se osigurava da je aplikacija dostupna, funkcionalna i prenosiva preko više različitih operativnih sustava i web preglednika.

Platforma je implementirana na način da može biti korištena kao zasebna web aplikacija koja se spaja na udaljeni poslužitelj ili kao desktop aplikacija koja automatski u pozadini pokreće poslužitelj i omogućava korisniku da koristi aplikaciju bez posebno pokretanja i postavljanja poslužitelja. Desktop i mobilna integracija je moguća kroz korištenje tauri programskog okvira.

Tauri je programski okvir za izgradnju malih, brzih programa za sve desktop i mobilne platforme. Aplikacija može integrirati bilo koji frontend okvir koji se kompajlira u HTML, javascript, css i koristiti ove tehnologije za razvoj korisničkog sučelja, a istovremeno može koristiti jezike poput rusta, swifta, jave i kotlina za backend logiku i interakciju s hardverom [8]. U ovom projektu tauri služi za pakiranje projekta u jednu izvršnu datoteku, te jednostavno postavljanje i pokretanje poslužitelja i jednog klijenta.

Aplikacija se sastoji od dva odvojena dijela, to su poslužitelj i klijent. Poslužitelj služi za spremanje podataka, spajanje i sinkronizaciju podataka između povezanih klijenata, te učitavanje slika, kako bi bile dostupne klijentima preko web API-a, te kreiranje manjih sličica (engl. *thumbnail*) iz učitanih slika, da klijenti ne bi trebali preuzimati cijele slike koje mogu biti veće rezolucije, a nikada neće biti potrebna takva rezolucija. Klijent sadrži svo korisničko sučelje i logiku igre, komunicira kroz poslužitelja s ostalim klijentima te prima promjene o stanju igre od poslužitelja.

Cijela aplikacija je umrežena, sve promjene koje se događaju kod jednog klijenta budu sinkronizirane i prikazane kod ostalih klijenata koji su povezani u istu sesiju. Aplikacija koristi ravnopravne čvorove (engl. *peer-to-peer*) za umrežavanje zbog jednostavnosti implementacije i veće fleksibilnosti, prije implementacije u obzir je došlo umrežavanje s autoritativnim poslužiteljem, više o oba pristupa i zašto je odabrano peer-to-peer umrežavanje u poglavlju 4.2.6.

4.1. Web aplikacije

Web aplikacija je program klijent-poslužitelj arhitekture koji koristi web preglednik kao svog klijenta. Web aplikacija obavljanje interaktivnu uslugu povezivanja s poslužiteljem putem interneta ili intraneta [9].

Razlikuje se od tradicionalne web stranice na slijedeće načine [9]:

- **Dinamičko naspram statičkog**: dok web stranica, u većini slučajeva isporučuje sadržaj statičkih datoteka, web aplikacija prikazuje dinamički prilagođen sadržaj na temelju parametara zahtjeva, korisničke sesije.
- **Interaktivnost**: web aplikacije omogućuju korisnicima obavljanje specifičnih zada-

taka, poput kupnje robe ili upravljanje zalihama.

Web aplikacije se temelje na distribuiranoj arhitekturi koja se sastoji od tri glavna elementa. Prvi element je klijent još nekad i nazivan frontend; to je obično web preglednik instaliran na korisnikovom računalu. Preglednik je odgovoran za prikazivanje sadržaja i rukovanje interakcijama korisnika. Drugi element je poslužitelj, ponekad zvan backend; ovaj element sluša dolazne zahtjeve od klijenata i generira odgovore. Obrađuje zadatke poput posluživanja statičkih datoteka, autentifikacije i prosljeđivanja dinamičkih zahtjeva aplikacijskim programima. Treći element je baza podataka; web aplikacije često zahtijevaju pristup trajnim podacima. Povezuju se s bazama podataka kako bi pohranile i dohvaćale informacije. Arhitektura aplikacije uključuje specifičnu logiku za omogućavanje pristupa podacima i upravljanje transakcijama [10].

4.2. Poslužitelj

U ovom potpoglavlju prolazimo kroz implementaciju poslužitelja, tehnologije koje su korištene i odluke koje su donesene vezane uz dizajn cijele aplikacije, kao što je odluka za implementaciju peer-to-peer umrežavanja umjesto autoritativnog poslužitelja.

Poslužitelj je implementiran u programskom jeziku rust. Rust je sistemski programski jezik sa statičnim tipovima podataka, dizajniran za performanse i sigurnost. Razvijen od strane Mozilla Research, stvoren je kao odgovor na dugogodišnju napetost između kontrole na niskoj razini koju pruže jezici poput C i C++-a i sigurnosti jezika više razine poput jave ili pythona. Rust sigurnost postiže sustavom zvan vlasništvo (engl. *ownership*) i provjeravanjem vlasništva nad varijablama tokom kompajliranja programa sustavom zvanim provjeravač posuđenog (engl. *borrow checker*). Korištenjem ova dva sustava osigurava korektan pristup i dijeljenje memorije, izbjegava probleme s korištenjem izbrisane memorije, čitanja ili pisanje u memoriju koju više dretva istovremeno koristi, a pristup memoriji nije sinkroniziran [11].

Budući da rust nema sakupljača smeća (engl. *garbage collector*) i ima minimalan runtime, nevjerojatno je brz i memorijski učinkovit. Kod kompajliranja programa koristi apstrakcije bez troška, što znači da programer može koristiti koncepte programiranje više razine, poput iteratora ili generičnih funkcija, bez da računalo obavlja dodatni posao tijekom izvođenja, svi koncepti više razine tokom kompajliranja su pretvoreni u nižu razinu, na primjer iteratori su pretvoreni u jednostavne for petlje, dok su generične funkcije pretvorene u više različitih funkcija koje kompajlirani program poziva [11].

4.2.1. Protokoli za prijenos podataka između poslužitelja i klijenta

Implementirani poslužitelj koristi http i websockete za komunikaciju s klijentima. HTTP je protokol na kojim se temelji world wide web, dizajniran je za prijenos specijaliziranih poruka putem mreže. Koristi TCP kao svoj temeljni transportni sloj, što osigurava da su svi podaci preneseni u točnom redoslijedu i primljeni. Komunikacija http-a koristi model zahtjeva i odgovora. HTTP klijent poput web preglednika započinje komunikaciju slanje poruke zahtjeva prema http

poslužitelju. Poslužitelj zatim zaprima zahtjev, obrađuje ga i generira poruku odgovora koju šalje natrag klijentu. Na ovaj način klijent može zatražiti bilo kakvu vrstu resursa od poslužitelja [10].

HTTP zahtjev se sastoji od nekoliko dijelova i počinje s metodom zahtjeva, putanju do resursa i broj verzije http-a, slijedeći redovi su popis zaglavlja, nakon zaglavlja je prazni redak iza kojeg se u poruku dodaje tijelo. Tijelo je opcionalno i ne mora biti prisutno u poruci. Ispod je primjer zahtjeva, korištena je metoda GET, pokušava se dohvatiti resurs index.html, verzija HTTP-a je 1.1 i u zahtjev uključeno je jedno zaglavlje, tijelo je prazno. Ovakvi zahtjevi su uglavnom slani s klijenta prema poslužitelju, no nije neubičajeno da dva poslužitelja komuniciraju preko http-a, te u ovom slučaju jedan poslužitelj preuzima ulogu klijenta.

```
GET /index.html HTTP/1.1
User-Agent: Mozilla/5.0(X11;Linux x86_64) Chrome/143.0.0.0
```

Klijent od poslužitelja prima odgovor na zahtjev. Struktura odgovora počinje statusnom linijom, koja sadrži verziju http-a, troznamenkasti statusni kod i kratko objašnjenje poslanog koda, npr. OK. U odgovoru, jednako kao kod zahtjeva, nakon prve linije dolazi lista zaglavlja koja su poslana, te nakon zaglavlja prazni redak i tijelo odgovora. U slučaju ispod tijelo je popunjeno i klijent prima html dokument koji može prikazati u web pregledniku.

```
HTTP/1.1 200 OK
Content-Type: text/html
Content-Lenght: 40

<html><h1>Hello World!</h1></html>
```

HTTP protokol je protokol bez stanja, što znači da je životni vijek veze između klijenta i poslužitelja ograničen na jednu razmjenu zahtjeva i odgovora. Poslužitelj ne održava stanje veze tijekom prijenosa uzastopnih naredbi. Ne može pamtiti niz interakcija niti grupirati zahtjeve zajedno. Budući da protokol ne održava informacije o sesiji, web aplikacije moraju koristiti druge mehanizme, poput kolačića za uspostavljanje trajnih sesija ili stanja kroz višestruke zahtjeve [10].

Websocket je protokol koji omogućuje dvosmjernu komunikaciju između klijenta i poslužitelja preko jedne, dugotrajne TCP veze. Za razliku od klasičnog http-a koji koristi model zahtjev odgovor, websocket omogućuje da obje strane šalju poruke u bilo kojem trenutku, bez potrebe za novim zahtjevima [12]. Ova karakteristika je idealna za aplikacije koje zahtijevaju ažuriranja u stvarnom vremenu i možemo nazvati websocket kao protokol koji održava stanje, za razliku od http-a koji je bez stanja. Uspostava websocket veze počinje kao običan http zahtjev koji je nadograđen na websocket protokol. Klijent šalje posebni http GET zahtjev s zaglavljem `Upgrade: websocket`. Poslužitelj odgovara statusom `101 Switching Protocols` ako prihvaća zahtjev. Nakon ovoga, veza prelazi na websocket protokol. Poruke se šalju u obliku okvira i postoji nekoliko vrsta okvira, to su tekst okviri, za slanje tekstualnih poruka, bi-

narni okviri, za slanje biranih poruka i kontrolni okviri, za slanje keep-alive poruka i zatvaranje veze [13]. U kasnijim poglavljima su prikazani primjeri gdje i kako se oba protokola koriste.

4.2.2. Axum

Axum je okvir za izradu web aplikacija u ekosustavu programskog jezika rust, dizajniran za izgradnju asinkronih http usluga s naglaskom na ergonomiju, modularnost i kompozabilnost. Pruža visokorazinsku apstrakciju za usmjeravanje, obradu zahtjeva i generiranje odgovora, dok koristi nižerazinske komponente iz asinkronog rusta. Naglašava tipksi sigurno usmjeravanje bez korištenja makro komandi i mehanizme ekstrakcije, omogućujući deklarativno parsiranje zahtjeva putem ekstraktora, na primjer pruža ekstraktore za parametre puta ili parsiranje JSON tijela zahtjeva [14].

Axum se u aplikaciji koristi kao web poslužitelj, odgovoran je za povezivanje klijenata, spajanje s bazom podataka, pohranjivanje podataka o igrama, te upravljanje i serviranje slika koje klijenti pohrane za neku igru. Kroz nekoliko primjera koda ćemo proči glavne značajke axum-a i način na koji se koristi kroz aplikaciju.

```
1  pub fn get_router<T: AppStateTrait>(state: T) -> axum::Router {
2      axum::Router::new()
3          .route("/assets/upload", routing::post(upload::upload))
4          .route(
5              "/assets/thumbnails/{image_id}",
6              routing::get(serve::thumbnails),
7          )
8          .route("/assets/{filename}", routing::get(serve::serve_file))
9          .layer(DefaultBodyLimit::disable())
10         .layer(RequestBodyLimitLayer::new(FILE_SIZE_LIMIT))
11         .with_state(state.clone())
12 }
```

Isječak koda 1: Kreiranje axum ruta za učitavanje i serviranje slika

U isječku koda 1 je prikazana funkcija koja kreira axum `Router` objekt s globalnim stanjem aplikacije `state`. U funkciji su definirane tri različite rute, detalji o rutama su u poglavlju 4.2.5. Funkcija `route` kao argumente prima putanju na koju će web poslužitelj reagirati ako je zatražena od klijenta, drugi argument registrira http metodu i funkciju koja se poziva za generiranje odgovora na klijentov zahtjev. Na primjer `route("/assets/upload", routing::post(upload::upload))` registrira rutu `"/assets/upload"`, definira da prima samo zahtjev za `POST` metodu, te ako klijent napravi ispravan zahtjev na ovu rutu poziva funkciju `upload::upload`. Funkcija `with_state` prima dijeljeno globalno stanje aplikacije i omogućava funkcijama koje generiraju odgovor klijentu pristup tome stanju, na primjer stanje može sadržavati vezu na bazu podataka. U slijedećem isječku je prikazana struktura globalnog dijeljenog stanja.

```rust
pub struct AppStateConfig<F>
where
    F: Adapter,
{
    pub file_system_handler: F,
    pub database: DatabaseConnection,
    pub entity_queue: Arc<Mutex<EntityQueue>>,
}

impl AppStateConfig<local_adapter::Local> {
    pub async fn get_default_config() -> AppStateConfig<local_adapter::Local> {
        let database = Self::get_database().await;
        let entity_queue = Arc::new(Mutex::new(EntityQueue::new(database.clone())));
        Self {
            file_system_handler: Self::get_fs_handler_from_config(),
            database,
            entity_queue,
        }
    }
}

impl<F> AppStateTrait for AppState<F> where F: Adapter
{
    type FsHandler = F;

    fn get_fs_handler(&self) -> Self::FsHandler { self.fs_handler.clone() }
    fn get_db(&self) -> DatabaseConnection { self.database.clone() }
    fn get_entity_queue(&self) -> Arc<Mutex<EntityQueue>> {
    ↪  self.entity_queue.clone() }
    fn get_scheduler(&self) -> Scheduler { self.scheduler.clone() }
}
```

Isječak koda 2: Implementacija dijeljenog globalnog stanja

Zbog jednostavnosti aplikacija koristi jedno globalno stanje, iako axum podržava različito stanje za svaki `Router` objekt. Globalno stanje se sastoji od objekta za upravljanje datotekama, te u ovom slučaju stanje koristi implementaciju za pristup datotekama na lokalnom računalu tj. pristup datoteka na računalu koje pokreće poslužitelj. Stanje se također sastoji od veze s bazom reda čekanja entiteta `EntityQueue`. Red čekanja entiteta će biti detaljnije objašnjen u kasnijem poglavlju, no za sad je samo bitno znati da postoji u globalnom stanju i bude dostupan kroz ekstraktore u funkcijama za generiranje odgovora na zahtjeve klijenta.

```rust
pub async fn upload<F: Adapter>(
    asset_manager: AssetManager<F>,
    conn: DbConn,
    multipart: extract::Multipart,
) -> Result<Json<UploadResponse>> {
    let file = UploadedFile::from_multipart(multipart).await?;

    let transaction = conn.begin().await?;

    let asset = asset_manager
        .create(
            &transaction,
            file.name.to_string(),
            &file.data,
            AssetType::File,
        )
        .await?;

    let thumbnails = if infer::is_image(&file.data) {
        asset_manager
            .create_thumbnail_assets(&transaction, &asset, Some(&file.data))
            .await?
    } else { vec![] };

    transaction.commit().await?;

    Ok(Json(UploadResponse {
        id: asset.id,
        thumbnails: thumbnails
            .into_iter()
            .map(|t| gen_partial_asset_url(&t.name))
            .collect(),
        url: gen_partial_asset_url(&asset.name),
        filename: asset.name,
        original_filename: asset.original_filename,
    }))
}
```

Isječak koda 3: Implementacija upload funkcije za učitavanje datoteka na poslužitelja

Isječak 3 prikazuje cijelu implementaciju funkcije za učitavanje datoteke s klijenta na poslužitelja. U isječku 2 definirano je definirana struktura globalnog stanja koje aplikacija koristi, a u ovoj funkciji se koriste dva objekta iz izvedena iz tog stanja. Ti objekti su `AssetManager<F>` i `DbConn`, dok axum pruža implementaciju za `extract::Multipart`. Funkciju u ni jednom trenutku ne pozivamo ručno, već kroz registraciju rute i korištenjem rust kompajlera axum generira poziv na definiranu funkciju s točnim parametrima. To radi na način da svi parametri funkcije moraju implementirati trait `FromRequestParts`, što omogućava axum-u mogućnost izvlačenja, postavljanja i prilagođavanje parametara.

Nadalje u funkciji se axum-ov ekstraktor `extract::Multipart` koristi za dobivanje podataka o datoteci koja se šalje poslužitelju, nakon izvlačenja podataka pokreće se transakcija

na razini baze podataka. Transakcija osigurava da će se u bazu zapisati podaci o učitanom dokumentu samo ako sve prođe bez greške. Ako se u bilo kojem trenutku kod pisanja podataka u bazu ili kasnijeg generiranja sličica originalne datoteke dogodi greška od koje se aplikacija ne može oporaviti, transkacija će biti obustavljena i podaci neće biti zapisani u bazu podataka. Kod završetka funkcije generira se odgovor koji se šalje klijentu. Odgovor je u JSON formatu, format je vidljiv kroz tip podataka koji funkcija vraća `Json<UploadResponse>`.

```rust
pub struct DbConn(DatabaseConnection);
impl<S> FromRequestParts<S> for DbConn
where
    S: AppStateTrait,
{
    type Rejection = Error;

    async fn from_request_parts(
        _parts: &mut Parts,
        state: &S,
    ) -> core::result::Result<Self, Self::Rejection> {
        Ok(state.get_db().into())
    }
}
```

Isječak koda 4: Implementacija ekstraktora za vezu s bazom podataka

Isječak iznad prikazuje implementaciju traita `FromRequestParts` koji omogućava axum-u, te na kraju programeru da koristi prilagođene ekstraktore u funkcijama za generiranje odgovora na klijentove zahtjeve. U isječku je definirano da varijabla `state` mora implementirati `AppStateTrait` (implementacija vidljiva u isječku 2) što garantira da varijabla `state` ima metodu `get_db` i može kreirati vezu s bazom podataka. Tu vezu dalje axum može proslijediti u funkcije koje koriste ekstraktor `DbConn`.

4.2.3. Socketioxide i Socket.IO

Socketioxide je implementacija socket.io poslužitelja [15] u rust-u. Biblioteka je prigodna za aplikaciju jer ima direktnu integraciju s axum-om. Socket.IO je biblioteka koja omogućava koja omogućuje događanjima vođenu dvosmjernu komunikaciju u stvarnom vremenu između web klijenata, obično web preglednika i poslužitelja. Namijenjen je aplikacijama koje zahtijevaju trenutno slanje podataka između klijenta i poslužitelja, s minimalnim kašnjenjem. Za potrebe aplikacije socket.io je korišten za uspostavljanje websocket veze između klijenata i poslužitelja, te za prosljeđivanje podataka s jednog klijenta na drugi, bez da ti klijenti moraju biti direktno povezani jedni s drugima.

Komunikacija kroz socket.io je bazirana na slanju događaja između klijenta i poslužitelja. Da bi se neki dio koda izvršio primitkom određenog događaja, klijent ili poslužitelj trebaju imati registriran slušatelj (engl. *listener*) za taj događaj. Na primjer, u isječku koda 5 postoji linija `socket.on(JOIN_EVENT, join_handler::<T>);` koja označava da će poslužitelj reagirati na događaj `JOIN_EVENT` koji dobije od klijenta.

Isječak koda ispod prikazuje način na koji se socketioxide integrira s axum-om, kako se kreira namespace i ruta na koji se klijenti mogu spajati, te dodatne postavke vezane za spajanje i održavanje veze s klijentom, također opet je vidljivo globalno stanje aplikacije `AppStateTrait` koje se prosljeđuje u socketioxide kontekst kako bi i socketioxide imao pristup globalnom stanju aplikacije, u poglavlju 4.2.6 je konkretno objašnjeno gdje se, u ovom kontekstu, globalno stanje koristi. Prije registracije nove rute potrebno je kreirati i konfigurirati socketioxide uslugu koja se može proslijediti u rutu. Usluga ima postavljeno na šezdeset sekundi vremena čekanja za `ping` i `ack` događanja. Ovo su posebna događanja koja socketioxide koristi za provjeru veze s klijentom i uspješno dostavljenih poruka/događaja. Poslužitelj u nekom intervalu šalje ping događaj klijentu, te klijent ima šezdeset sekundi za poslati pong odgovor, u suprotnom veza s klijentom bude isključena. Na sličan način funkcionira ack događaj, kod ovog događaja klijent ima određeno vrijeme za slanje odgovora primitka nekog događaja od poslužitelja. Ovim mehanizmima socket.io garantira da su klijenti aktivni, te ako nisu može isključiti vezu s njima i obustaviti slanje događaja prema neaktivnim klijentima.

`io.ns("/", on_connect::<T>);` definira da će se pozvati `on_connect` funkcija kad se klijent spoji na namespace `/`. Socket.io koristi namespace kao jedan od načina za grupiranje povezanih klijenata i odabir klijenti kojima će se slati događaji, drugi način su sobe, poslužitelj odlučuje ako i u koju sobu će pridružiti klijenta. Sobe su koncept jedino dostupan poslužitelju, klijenti nemaju pristup popisu soba kojima su pridruženi. Spajanjem klijenta poziva se funkcija `on_connect` koja služi da se povezanom klijentu registrira povratni poziv funkcije u slučaju ako se klijent isključi te postavlja vezu u određeno stanje. Drugi registrirani povratni poziv funkcije se koristi za povezivanje klijenta u određenu sobu i slanje podataka o igri tom klijentu, te povezivanje tog klijenta s drugima u istoj sobi. Sve dok klijent ne pošalje odgovarajući `JOIN_EVENT` događaj neće primati nikakve događaje od poslužitelja, ni poslužitelj neće primati nikakve druge događaje od klijenta. Više o povezivanju između klijenata i poslužitelja u poglavlju 4.2.6.

```rust
pub fn get_router<T: AppStateTrait>(state: T) -> axum::Router {
    let (service, io) = SocketIoBuilder::new()
        .with_state(state.clone())
        .ping_timeout(Duration::from_secs(60))
        .ack_timeout(Duration::from_secs(60))
        .connect_timeout(Duration::from_secs(60))
        .build_svc();

    io.ns("/", on_connect::<T>.with(auth_middleware::<T>));

    axum::Router::new().route_service(
        "/socket.io/",
        ServiceBuilder::new()
            .layer(CorsLayer::permissive())
            .service(service),
    )
}

pub fn on_connect<T: AppStateTrait>(socket: SocketRef) {
```

```
17      let socket_clone = socket.clone();
18      socket.on_disconnect(move |reason: DisconnectReason| {
19          socket_clone.extensions.remove::<JoinedFlag>();
20      });

21      socket.on(JOIN_EVENT, join_handler::<T>);
22  }
```

Isječak koda 5: Postavljanje socketioxide poslužitelja kao axum rutu

4.2.4. Baza podataka

Baze podataka koriste se u aplikacijama kako bi osigurale pouzda, trajan i strukturiran način pohrane informacija. Omogućuju organiziranje podataka u tablice i relacije, što olakšava upravljanje složenim skupovima podataka. Osim pohrane, baze podataka ključne su za očuvanje integriteta podataka putem transakcija, koje osiguravaju da podaci ostanu dosljedni čak i u slučaju kvarova sustava. Također omogućavaju aplikacijama pretraživanje i dohvaćanje specifičnih informacija putem upitnih jezika poput SQL-a [16].

U aplikaciji se koristi SQLite za pohranu podataka. SQLite je odabran zbog jednostavnosti direktnog ugrađivanja baze podataka u aplikaciju. Za razliku od tradicionalnih baza podataka, popust PostgreSQL-a, koji koriste klijent-poslužitelj arhitekturu i zahtjevaju zaseban poslužiteljski proces za upravljanje podacima i zahtjevima, SQLite integrira cijeli mehanizam baze podataka izravno u aplikaciju. SQLite cijelu bazu podataka pohranjuje u jednu datoteku [16].

Bez posebne konfiguracije SQLite nije najbolje namijenjen za istovremeno pisanje i čitanje podataka. Zato su u slijedećem isječku koda prikazane korištene postavke za SQLite vezu, kako bi se poboljšale performanse kod istovremenog pisanja i čitanja za više klijenata. `journal_mode` je postavljena na `WAL` kako bi se omogućilo istovremeno čitanje i pisanje, `synchronous` opcija je postavljena na `NORMAL` kako bi se izbjegao gubitak podataka u slučaju da se dogodi neka greška pri pisanju podataka iz write ahead log-a (WAL-a) u bazu. Opcija `busy_timeout` ima vrijednost od pet sekundi, te ova opcija omogućava bazi da pričeka određeno vrijeme prije vraćanja greške u slučaju da pokušava zapisivati u redove zaključane drugom transakcijom. Opcija `cache_size` povećava veličinu predmemorije i smanjuje broj zapisa koji će se raditi na disk. `foreign_keys` uključuje provjeru stranih ključeva na tablicama, bez ove opcije SQLite ne provjerava ako tablica ima relaciju na neku drugu tablicu, ni ako je ta relacija validna. Postavljanjem `auto_vacuum` na `INCREMENTAL` omogućuje postepeno oslobađanje prostora na disku kako se retci iz baze brišu, umjesto potpunog oslobađanja tijekom nekih od operacija nad bazom. Postavljanjem `temp_store` na `MEMORY` SQLite će privremene tablice pohraniti u memoriji umjesto spremati ih na disk i kasnije odbaciti [17].

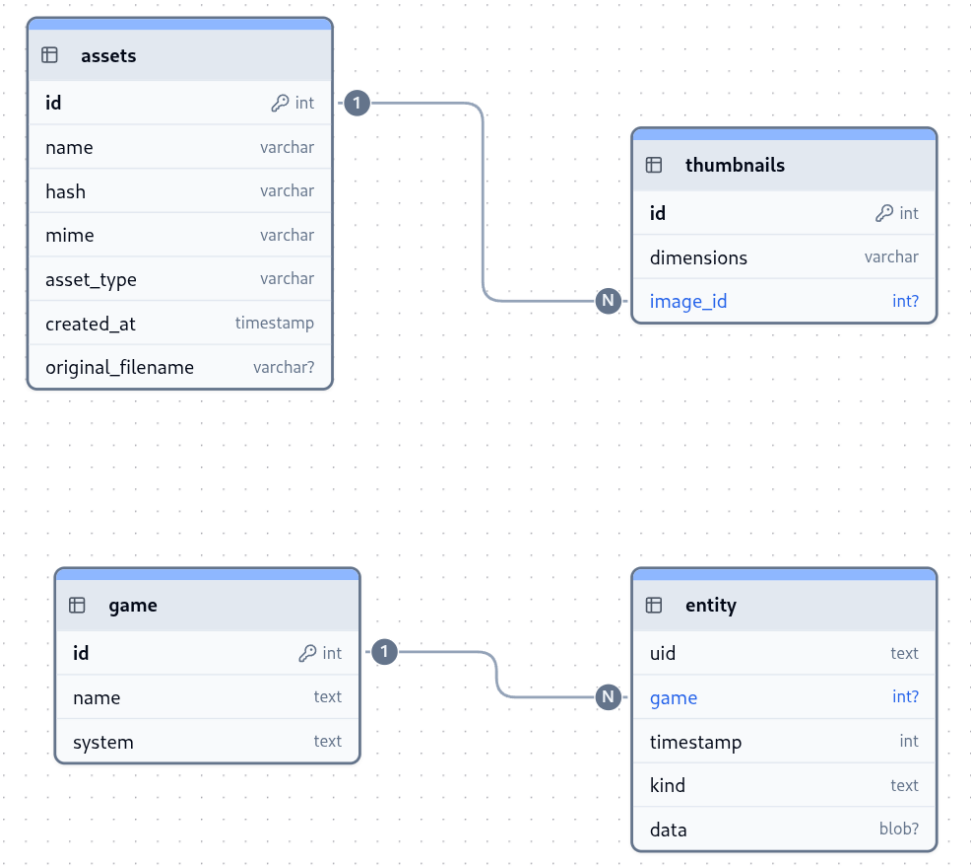
```
1  let pool = db.get_sqlite_connection_pool().clone();

2  let connect_options = (*pool.connect_options())
```

```
    .clone()
    .journal_mode(sqlx::sqlite::SqliteJournalMode::Wal)
    .synchronous(sqlx::sqlite::SqliteSynchronous::Normal)
    .busy_timeout(std::time::Duration::from_secs(5))
    .pragma("cache_size", "-20000")
    .foreign_keys(true)
    .auto_vacuum(sqlx::sqlite::SqliteAutoVacuum::Incremental)
    .pragma("temp_store", "MEMORY");
pool.set_connect_options(connect_options);
```

Isječak koda 6: Postavljene SQLite opcije

Na slijedećoj slici je prikaz svih tablica i relacija u bazi podataka. Baza se sastoji od tablica `assets`, `thumbnails`, `game` i `entity`. Tablica `assets` definira datoteku učitanu na poslužitelja, ima nekoliko stupaca koji opisuju naziv, vrstu datoteke, originalni naziv datoteke. `Thumbnails` tablica ima relaciju na `assets` tablicu, jedan red u thumbnails tablici predstavlja sličicu koja je generirana za originalnu učitanu sliku. Tablica `game` samo zadrži naziv igre i sustav koji igra koristi, dok `entity` tablica sadrži sve entitete vezane za neku igru i sve njihove podatke. Podaci i vrsta entiteta definirana je od strane klijenta tj. implementacije igre, jednako tako klijent je odgovoran za spremanje id-a asset-a u neki od podataka u entitetu, poslužitelj nema način za pregled svih asset-a i provjeru kojoj igri pripadaju.



Slika 5: Shema baze podataka [autorski rad]

4.2.5. API krajnje točke

U tablici ispod su definirane sve krajnje točke kojima klijenti imaju pristup i na koje poslužitelj odgovara. Većina komunikacije između poslužitelja i klijenta odvija se kroz websocket. Može se primijetiti da u popisu fali krajnja točka za pridruživanje igri, fali iz razloga što se pridruživanje izvršava kroz websocket te ima nekoliko koraka koje klijent mora odraditi da bi se uspješno pridružio igri. Više o načinu pridruživanja u poglavlju 4.2.6.

Tablica 4: API krajnje točke [autorski rad]

Ruta	**HTTP Metoda**	**Opis**
/public/{path}	GET	Omogućava dohvaćanje datoteka koji su dostupni javnosti, na primjer ikonica ili css stilova
/socket.io/	GET	Služi za povezivanje socket.io klijenata
/assets/upload	POST	Omogućuje učitavanje datoteka na poslužitelj. Poslužitelj odgovara s idem datoteke linkovima i id-evima sličica ukoliko je učitana datoteka slika
/assets/thumbnails/{image_id}	GET	Dohvaća sličicu definiranu parametrom `image_id`
/assets/{filename}	GET	Dohvaća dokument definiran parametrom `filename`
/game/list	GET	U `JSON` formatu šalje popis svih dostupnih sesija igre
/game/create	POST	Kreira novu sesiju igre

4.2.6. Mrežna komunikacija

Mrežna komunikacija između klijenta i poslužitelja se odvija na dva načina. Prvi je kroz API krajnje točke, definirane u poglavlju 4.2.5, drugi je kroz websocket koristeći socket.io. U isječku koda 5 je vidljivo korištenje funkcije `.with(auth:middleware::<T>)`, da bi se klijent uspješno povezao na websocket mora poslati određeni sadržaj pri povezivanju na `/socket.io/` krajnju točku. Sadržaj koji klijent mora poslati je `id` igre u koju se želi pridružiti. Poslužitelj će tada obaviti autentifikaciju i korištenjem `socket.extension.insert(auth);` zapisati podatke na instancu websocketa kako bi se podaci mogli kasnije koristiti, te da bi poslužitelj imao uvid u koju igru je klijent pridružen.

```
1 fn auth_middleware<T: AppStateTrait>(
2     socket: SocketRef,
3     State(app_state): State<T>,
4     Data(auth): Data<WebsocketAuthMessage>,
5 ) -> Result<(), websocket_auth::Error> {
```

```
    match auth.authenticate(&app_state) {
        Ok(()) => {
            socket.extensions.insert(auth);
            Ok(())
        }
        Err(e) => Err(e),
    }
}
```

Isječak koda 7: Autentifikacije websocket klijenta

U poglavlju 4.2.3 je definirano da nakon uspješnog spajanja klijenta, da će poslužitelj samo reagirati na događaja odspajanja klijenta s veze i događaj `JOIN_EVENT`. Kad je klijent spreman može poslati `JOIN_EVENT` događaj na poslužitelj. Poslužitelj će u trenutku primitka ovog događaja početi proceduru za pridruživanje klijenta u igru. U isječku koda 8 je vidljiva cijela procedura pridruživanja klijenta u igru, započinje s čitanjem id-a igre kojoj se klijent pridružuje. Podatak je isčitan s websocket instance koja je u trenutnu autentifikacije imala postavljene podatke o igri. Nakon toga poslužitelj pridružuje websocket u sobu. Sobe se koriste za slanje podataka između klijenata koji su povezani u istu igru. Također odmah nakon pridruživanja sobe, poslužitelj registrira slušatelj za događaj `ACTION` na websocket. U isječku je vidljivo ponovno korištenje `AppStateTrait` globalnog stanja za dohvaćanje reda čekanja entiteta i pokretanje transakcije u bazi podataka. Red čekanja je korišten za periodično spremanje entiteta u bazu podataka, te kod svakog pridruživanja se dohvaća i sprema u bazu kako bi kod dohvaćanja entiteta svi bili zapisani u bazi podataka. Nakon dohvaćanja svih entiteta iz baze, poslužitelj u manjim dijelovima šalje događaje `JOIN_EVENT` klijentu s podacima o entitetima i stanju slanja. Klijent je odgovoran za pretvaranje entiteta u objekte koje razumije i može koristiti tokom igre. Po završetku slanja entiteta poslužitelj šalje klijentu događaj `JOIN_FINISHED_EVENT` što označava da su svi entiteti uspješno poslani, te klijent ima sve podatke za pridruživanje i sinkronizaciju igre.

```
#[tracing::instrument(skip(socket, app_state))]
async fn join_handler<T: AppStateTrait>(
    socket: SocketRef,
    State(app_state): State<T>,
    Extension(auth): Extension<WebsocketAuthMessage>,
) {
    let game_id = auth.game;
    let room = format!("room-{game_id}");

    socket.join(room);
    socket.on(ACTION, action_handler::<T>);

    let queue = app_state.get_entity_queue();
    {
        let mut lock = queue.lock().await;
        if let Some(handle) = lock.flush().await { handle.await.ok(); }
    }
```

```
16      tracing::debug!("Starting database entity fetch");
17      let db = app_state.get_db();
18      let transaction = db.begin().await?;

19      let entity_manager = EntityManager::new();

20      let entities = entity_manager.load_entities(&transaction, game_id).await?;

21      let total = entities.len();
22      let mut sent = 0;
23      tracing::debug!("Sending entities in {} chunks", total / CHUNK_SIZE);
24      entities.chunks(CHUNK_SIZE).for_each(|chunk| {
25          sent += chunk.len();
26          socket
27              .emit(JOIN_EVENT, &serde_json::json!({
28                      "progress": {
29                          "sent": sent,
30                          "total": total
31                      },
32                      "data": chunk
33                  }),
34              )
35              .ok();
36      });

37      socket.emit(JOIN_FINISHED_EVENT, &()).ok();
38  }
```

Isječak koda 8: Procedura za pridruživanje

Pridruženi klijenti komuniciraju kroz poslužitelja slanjem događaja `ACTION`.

Koristi se peer to peer komunikacija, postoji server authoritive...

Naivna implementacija mrežne komunikacije između klijenata

4.3. Klijent

4.3.1. Peer to peer komunikacija

4.3.2. Stateless & stateful komunikacija

5. Korišteni alati i tehnologije za implementaciju

Aplikacija je podijeljena na dva dijela, na poslužitelja i klijenta. Poslužitelj se koristi za:

- kreiranje igara,
- povezivanje klijenata; sustav je napravljen s umrežavanjem klijenata, više klijenata mogu igrati jednu igru i pratiti događanja,
- spremanje i učitavanje podataka o aktivnim igrama; promjene koje klijenti naprave kroz igru moraju biti spremljene za kasnije učitavanje i sinkronizaciju ostalih klijenata,

Opisati svaki alat koji se koristi. Navesti u kratko gdje se koristi. Navesti da budu točni primjeri kasnije u radu.

5.1. Poslužitelj

5.1.1. Tauri

5.1.2. Axum

5.2. Baza podataka

5.3. Klijent

5.3.1. Pixi.js

5.3.2. React.js

5.3.3. Jotai

5.3.4. Socket.io

5.3.5. Mantine

5.4. Primjer borbe

Ovo je glavni dio rada u kojem treba razraditi temu, pojasniti istraživanja, prikazati rezultate i slično. Poželjno je na početku poglavlja dati kratki opis strukture poglavlja, kako bi čitatelj/čitateljica rada mogao/mogla lakše pratiti složenu cjelinu.

5.5. Poglavlje druge razine

5.5.1. Poglavlje treće razine

5.5.1.1. Poglavlje četvrte razine

6. Prikaz slučajeva korištenja

Tehničke upute u nastavku opisuju način tehničkog oblikovanja rada i navođenja literature.

6.1. Upute za oblikovanje izgleda rada

Stranice se oblikuju korištenjem sljedećih parametara:

- veličina i oblik papira je A4, okomito usmjerenje, margine 2,5 cm na svakoj strani;
- naslovna stranica rada se ne numerira;
- nakon naslovne stranice, sve sljedeće stranice do 1. Poglavlja se numeriraju rimskim brojevima, počevši od i;
- od 1. poglavlja nadalje, stranice se numeriraju arapskim brojevima;
- broj stranice treba pozicionirati desno 1,25 cm od dna stranice, font Arial 9.

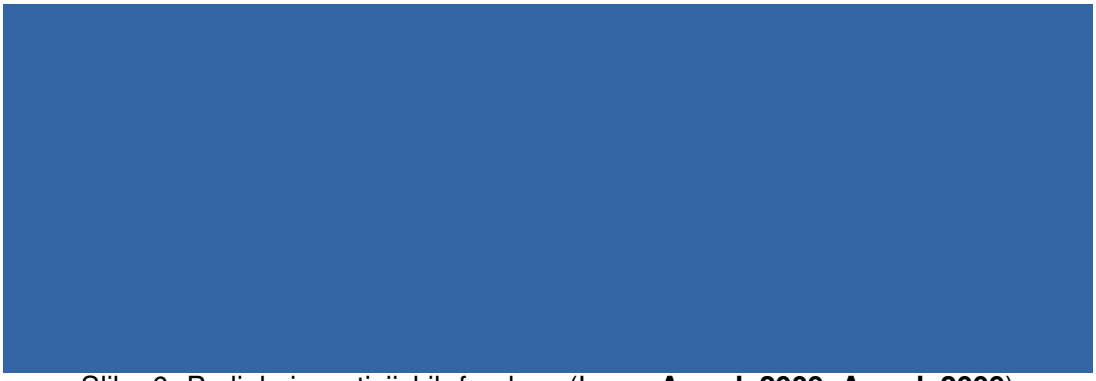
Tekst rada je potrebno oblikovati sukladno ovom predlošku, odnosno na sljedeći način:

- u pisanju teksta koristite font Arial 11 pt, s proredom 1,5 te razmakom 0 pt prije i razmakom 6 pt poslije odlomka, pri čemu je prvi redak uvučen za 1,25 cm;
- u naslovima prve razine „3. Razrada teme“ koristite font Arial 18 pt, podebljano, prijelom stranice (svaki naslov prve razine treba biti na novoj stranici), s proredom 1,5 te razmakom 0 pt prije i razmakom 18 pt poslije odlomka;
- u naslovima druge razine „2.1. Naslov“ koristite font Arial 16 pt, podebljano, s proredom 1,5 te razmakom 18 pt prije i razmakom 12 pt poslije odlomka;
- u naslovima treće razine „2.1.1. Naslov“ koristite font Arial 14 pt, podebljano, s proredom 1,5 te razmakom 12 pt prije i razmakom 6 pt poslije odlomka;
- u naslovima četvrte razine „2.1.1.1. Naslov“ koristite font Arial 12 pt, podebljano, s proredom 1,5 te razmakom 6 pt prije i razmakom 6 pt poslije odlomka;
- ostalo značajno isticanje cjelina rada može biti istaknuto podebljanim i kurziv slovima, korištenjem fonta Arial 11 pt.

Slike u radu je potrebno oblikovati na sljedeći način: naziv slike navedite ispod slike uz numeraciju;

- za nazive slika koristite iste postavke fonta kao i za tekst, ali stavite naziv slike u centrirani položaj;

- za oblikovanje same slike koristite font Arial 9 pt za tekst na slici; ispred same slike umetnite jedan prazan redak (osim ako je slika pozicionirana na početku stranice);
- nakon naziva slike ostavite jedan redak prazan (osim ako je naziv slike zadnji redak na stranici);
- kod prijeloma stranice treba obratiti posebnu pozornost da naziv slike, izvor i sama slika moraju biti na istoj stranici;
- slike je potrebno numerirati redom pojavljivanja u tekstu;
- ako je slika preuzeta iz drugog izvora, nakon navođenja naziva slike u zagradi navedite izvor, npr. (autor/autorica, godina);
- dozvoljeno je napraviti vlastitu preradu slika, grafikona ili tablica na način da se zadrži isti smisao sadržaja, ali promijeni izgled. I u takvim se slučajevima obavezno u nazivu navodi referenca izvornog djela ovako: “(Prema: Klačmer Čalopa i Cingula, 2012)“;
- dozvoljeno je preuzeti samo jednu sliku, grafikon ili tablicu u izvornom obliku iz istog izvora. Za doslovno preuzimanje većeg dijela sadržaja potrebno je ishoditi dozvolu nositelja autorskih prava;
- primjer označavanja slike možete vidjeti u nastavku (slika 6).



Slika 6: Podjela investicijskih fondova (Izvor: **Aranda2009**, **Aranda2009**)

Tablice rada je potrebno oblikovati sukladno ovim uputama:

- naziv tablice navedite iznad slike;
- za nazive tablica koristite iste postavke fonta kao i za tekst, ali stavite naziv tablice u centrirani položaj;
- za oblikovanje same tablice koristite font Arial 9 pt za tekst u tablici;
- tablice numerirajte redom pojavljivanja u tekstu;
- prije naziva tablice umetnite jedan redak prazan (osim ako je naziv tablice prvi redak na stranici);

- nakon same tablice umetnite jedan prazan redak (osim ako je tablica pozicionirana na kraju stranice);
- kod prijeloma stranice treba obratiti posebnu pozornost da naziv tablice, izvor i sama tablica moraju biti na istoj stranici;
- ako je tablica preuzeta iz drugog izvora, nakon navođenja naziva tablice potrebno je navesti izvor, na isti način kako je opisano kod slika;
- primjer označavanja tablice možete vidjeti u nastavku (tablica 5).

Tablica 5: Prikaz podataka o učestalosti pojavljivanja objekta

(Izvor: **caragliu2011smart**, **caragliu2011smart**)

Programski kod

- za oblikovanje teksta koji je programski kôd koristite font Courier, veličine 10 pt, jednostruki prored, 6 pt iza odlomka, npr. HTML kôd dijela zaglavlja početne web stranice FOI weba:

```
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <link rel="shortcut icon" href="https://www.foi.unizg.hr/sites/default/files/
   favicon_0_1.ico" type="image/vnd.microsoft.icon" />
  <meta name="generator" content="Drupal 7 (http://drupal.org)" />
  <link rel="canonical" href="https://www.foi.unizg.hr/hr" />
  <link rel="shortlink" href="https://www.foi.unizg.hr/hr" />
  <!-- Set the viewport width to device width for mobile -->
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Dobro došli na FOI | FOI</title>...
</head>
```

Formule

- za unos formula koristite editor za formule u svom tekst procesoru.

Kratice

- ako želite koristiti kratice pojmova u tekstu, kad prvi put spominjete pojam potrebno je navesti puni naziv, a kraticu navesti u zagradi (npr. Informacijske i komunikacijske tehnologije, kraće IKT). Nakon toga možete koristiti kratice u tekstu. Poželjno je u naslovima koristiti pune nazive.

Strano nazivlje

- strano nazivlje se u tekstu navodi u zagradi, napisano *kurzivom*, nakon hrvatskog izraza, npr. Analiza društvene mreže (engl. *Social Network Analysis - SNA*).

6.2. Navođenje literature

Za navođenje literature u radu možete odabrati i koristiti jedan od sljedeća dva ponuđena stila: **APA** ili **IEEE** stil. Važno je samo dosljedno primjenjivati odabrani stil u cijelom radu.

U popisu literature potrebno je navesti svu literaturu i samo literaturu koju ste koristili u tekstu.

Uz svaku preuzetu tvrdnju potrebno je navesti njezin izvor, tj. referencu. Reference se u tekstu navode tako da se uz citirani tekst navede izvor, sukladno načinu propisanom odabranim stilom i FOI preporukama za citiranje i referenciranje **SchattenEtAl2016roadmap**.

7. Zaključak

Ovdje treba sažeto rezimirati najvažnije rezultate razrade teme rada. Potrebno je sažeto opisati što je predmet rada, koje su metode, tehnike, programski alati ili aplikacije korištene u razradi rada te koje su pretpostavke dokazane, a koje opovrgnute. Sadržajno, ono što se u uvodu rada najavljuje i kasnije je obuhvaćeno u samom radu, moralo bi biti opisano u zaključnom dijelu kroz rezultate rada.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Popis literature

[1] J. H. Mann, „Experimental evaluations of role playing.,” *Psychological Bulletin*, sv. 53, br. 3, str. 227, 1956.

[2] W. J. White, J. Arjoranta, M. Hitchens, J. Peterson, E. Torner i J. Walton, „Tabletop role-playing games,” *The Routledge Handbook of Role-Playing Game Studies*, Routledge, 2024.

[3] J. Peterson, *Playing at the World: A History of Simulating Wars, People and Fantastic Adventures, from Chess to Role-playing Games*. Unreason Press, 2012., ISBN: 9780615642048.

[4] P. Sidhu, M. Carter i J. P. Zagal, *Fifty Years of Dungeons & Dragons*. MIT Press, 2024.

[5] M. Mearls i J. Crawford, *Player's Handbook*. Wizards of the Coast, 2014.

[6] M. Mearls i J. Crawford, *Dungeon Master's Guide*, 5th. Renton, WA: Wizards of the Coast, 2014., ISBN: 978-0786965625.

[7] W. of the Coast, *Dungeons and Dragons Character Sheet*, `https://media.wizards.com/2016/dnd/downloads/5E_CharacterSheet_Fillable.pdf`, Pristupano 26.11.2025, 2014.

[8] *What is Tauri?* `https://tauri.app/start`, Pristupano 12.12.2025.

[9] L. Shklar i R. Rosen, *Web application architecture*. John Wiley & Sons, 2009., sv. 36.

[10] S. Casteleyn, F. Daniel, P. Dolog, M. Matera i dr., *Engineering web applications*. Springer, 2009., sv. 30.

[11] *Rust Programming Language*, `https://rust-lang.org/`, Pristupano 12.12.2025.

[12] V. Pimentel i B. G. Nickerson, „Communicating and displaying real-time data with web-socket,” *IEEE Internet Computing*, sv. 16, br. 4, str. 45–53, 2012.

[13] B. Gupta i M. Vani, „An overview of web sockets: The future of real-time communication,” *Int. Res. J. Eng. Technol. IRJET*, sv. 5, br. 12, 2018.

[14] *Axum*, `https://docs.rs/axum/latest/axum/`, Pristupano 13.12.2025.

[15] *Socketioxide*, `https://github.com/Totodore/socketioxide`, Pristupano 13.12.2025.

[16] J. Kreibich, *Using SQLite*. " O'Reilly Media, Inc.", 2010.

[17] *Sensible SQLite defaults*, `https://briandouglas.ie/sqlite-defaults/`, Pristupano 15.12.2025.

Popis slika

Popis tablica