

**FOM Hochschule für Oekonomie & Management Essen**  
**Hochschulzentrum Siegen**



Berufsbegleitender Studiengang  
Wirtschaftsinformatik, 5. Semester

**Projektdokumentation als Seminararbeit**  
**im Rahmen der Lehrveranstaltung**  
**Web Technologie**

über das Thema  
**Browser RPG-Adventure**

Betreuer: Daniel Bitzer

Autoren: Rico (Matrikelnummer 12345)  
Henning (Matrikelnummer 12345)  
Julian (Matrikelnummer 12345)

Abgabe: 13. Februar 2022

## Inhaltsverzeichnis

<b>1 Einleitung</b>	<b>1</b>
<b>2 Aufgabenbeschreibung</b>	<b>2</b>
<b>3 Anforderungen</b>	<b>3</b>
<b>4 Herangehensweise</b>	<b>5</b>
<b>5 Vorstellung der Ergebnisse</b>	<b>8</b>
5.1 Infrastruktur und Code-Entwicklung . . . . .	11
5.1.1 Infrastruktur . . . . .	11
5.1.2 Initialisierung der Django-Anwendung . . . . .	14
5.1.3 Datenbankmodell . . . . .	15
5.1.4 Hervorzuhebende Code-Entwicklungen . . . . .	18
5.2 Spieldesign . . . . .	23
5.2.1 Allgemeines . . . . .	23
5.2.2 Balancing . . . . .	26
5.2.3 Storytelling . . . . .	28
5.3 Grafiken und Animationen . . . . .	30
5.4 Frontend-Design . . . . .	34
<b>6 Reflektion</b>	<b>38</b>
<b>Anhang</b>	<b>40</b>
<b>Literaturverzeichnis</b>	<b>50</b>

## Abbildungsverzeichnis

1	Projektsizze vom 27.11.2021 . . . . .	5
2	Frühes UI-Layout Mockup . . . . .	7
3	Konzept: Kampfsystem vom 29.11.2021 . . . . .	7
4	Diagramm: Seitenaufbau und Spielablauf . . . . .	10
5	Konzept: Entwicklung, Codeverteilung, Produktivnahme . . . . .	11
6	Bildschirmfoto: Server Auslastung . . . . .	13
7	Datenbankmodell Entwurfsprozess . . . . .	15
8	Datenbankmodell final (Stand 10.02.2022) . . . . .	16
9	Bildschirmfoto der Spieleseite zum Entwicklungsstand per 29.12.2021 . .	22
10	Details den Klassen aus der Projektbesprechung vom 11.12.2021 . . . .	23
11	Entwurfszeichnung und Text für eine Charaktererstellungsseite (29.11.2021)	28
12	Vom Concept Art zur finalen Grafik . . . . .	30
13	Krita Standard-UI . . . . .	31
14	Ebenenkonzept für die Grafiken . . . . .	32
15	Krita Animation UI . . . . .	33
16	Frontend-Design Beispiel: Szenenlobby . . . . .	34
17	Anhang: Allererstes Konzept mit Pseudocode für die Spiel-Logik (23.11.2021) . . . . .	42
18	Anhang: Erstes Mockup zu einer möglichen UI des Spiels (27.11.2021) .	43
19	Anhang: Detailierterer Pseudocode für die Level-Lobby (30.11.2021) . .	44
20	Anhang: Entwurf einer UI für die Lebel-Lobby mit Details (30.11.2021) .	45
21	Anhang: Konzept zum Zeitstrahl des Countdowns in der Lobby-Logik (02.12.2021) . . . . .	46
22	Anhang: Dokumentation einer Projektbesprechung (05.12.2021) . . . .	47
23	Anhang: Projekt Besprechung mit gegenseitigem Update und Wechsel von Szenenlogik zu Kampfsystem für das RPG (11.12.2021) . . . . .	48
24	Anhang: Projekt Besprechung mit Ermittlung von restlichen ToDos, Auf- gabenverteilung, Meilenstein- und Terminplanung (06.01.2022) . . . . .	49

## 1 Einleitung

Das Projektteam, bestehend aus Julian Schäfer, Henning Beier und Rico Pursche, beschäftigt sich im Rahmen dieser Projektarbeit des Studienfachs Web-Technologie mit der Entwicklung eines browserbasierten Rollenspiels. Als Projekt erschien den Studenten ein Spiel am interessantesten, denn damit kann sich jedes der Teammitglieder identifizieren. Dies erschien wichtig, um die Motivation an der Arbeit über den gesamten Zeitraum der Entwicklung hochzuhalten. Außerdem konnte sich jeder an die Zeiten erinnern als er das erste Mal eines der klassischen 8-Bit/16-Bit Spiele der Achtziger- und Neunzigerjahre gespielt hat und schnell wurde klar, dass das Spiel sich am Stil dieses auch heute noch beliebten Genres orientieren soll.

Das Spiel sollte jedoch nicht zu einfach gehalten werden, um eine gewisse Komplexität in die Entwicklung zu bringen und einer Projektarbeit des geforderten Umfangs zu entsprechen. Deshalb geben die Rahmenbedingungen vor, dass das Spiel sowohl Einzelspieler- als auch Mehrspielerkomponenten enthalten und eine Charakterentwicklung in Form von z. B. Levelaufstiegen haben soll. Außerdem ist es wünschenswert, dass alle Teilnehmer am Spiel, die für sie relevanten Informationen vom Spiel und den anderen Spielteilnehmern in Echtzeit erhalten. Dabei sollen aktuelle und geeignete Technologien zum Einsatz kommen, die den Kenntnisstand und den individuellen Fähigkeiten der einzelnen Teammitglieder entsprechen und mit denen die oben genannten Kriterien, vor allem die Online-Fähigkeit, mit Hilfe eines Browsers zu spielen, erfüllt werden können.

So entstand die Idee ein browserbasiertes RPG im oben genannten Stil zu erschaffen und im Folgenden soll gezeigt werden, wie das Team dies im Detail umgesetzt hat. Welche Probleme bei der Spieleanwendung auftraten und wie diese gelöst wurden, soll dabei eben so behandelt werden wie die genaue technische Umsetzung einzelner Details.

## 2 Aufgabenbeschreibung

Am Ende der Spieleentwicklung soll als primäres Ziel der Projektarbeit ein RPG (Roleplaying Game) mit Fantasy-Setting entstehen, bei dem es vor allen Dingen um die Fertigstellung des kompletten Spielumfangs und nicht nur einiger Teilbereiche geht. Das Spiel soll also von Anfang bis Ende durchgespielt werden können, alle an das Projekt gestellten Anforderungen erfüllen und die gewünschten Inhalte bieten. Im Einzelnen sind im o.g. Zusammenhang folgende Aspekte zu nennen. Das Spiel muss mit einem gängigen Browser online spielbar sein, es soll sowohl Einzelspieler- als auch Mehrspielerlemente enthalten und es soll verschiedene Charakterklassen geben, die sich in ihren Fähigkeiten klar voneinander unterscheiden. Im Laufe des Spiels sollen sich, bei Erreichen bestimmter Grenzen in einem für jeden Charakter eigenem Erfahrungspools, diese Fähigkeiten verbessern. Des Weiteren wird es vom Projektteam unabhängig vom Endprodukt ebenfalls als primäres Ziel angesehen, den Projektablauf erfolgreich zu gestalten.

Als sekundäre Ziele, also den primären Zielen klar untergeordnet, sieht die Projektgruppe die Spielerfahrung der einzelnen Spieler. Es ist allerdings angedacht diese in den Rahmen der Möglichkeiten so intensiv wie möglich zu gestalten, jedoch steht wie oben bereits erwähnt die Umsetzung der Vorgaben im Vordergrund. Außerdem möchte jeder der Teilnehmer an diesem Projekt Erfahrung im Spieldesign sammeln.

Die Herausforderungen in diesem Projekt sind recht vielfältig und sind grob in technische und persönliche zu gliedern.

Zu Ersterem ist zu zählen, dass zu Beginn des Projekts nicht klar ist, ob das vom Team erst einmal theoretisch entwickelte Konzept auch real technisch umzusetzen ist. Zum Einen, weil nicht sicher ist, ob die gewählte technische Plattform geeignet ist und zum Anderen, weil nicht klar ist ob das Know-How des Teams reicht, um das Gewollte in ein fertiges Endprodukt zu gießen. Außerdem könnten Rechte an Grafiken und vielleicht sogar Programmcode zum Problem werden.

Die persönlichen Herausforderungen liegen darin, das ganze Projekt zeitlich so zu gestalten, dass es neben der hauptberuflichen Tätigkeit des Projektteams noch genug Zeit für die Umsetzung bleibt. Krankheit oder Quarantäne in der aktuellen Pandemie von einzelnen Teammitgliedern könnten den zeitlichen Rahmen ebenfalls in Gefahr bringen. Auch muss für jedes Mitglied des Teams ein geeignetes Aufgabenfeld gefunden werden, so dass die individuellen Fähigkeiten sich möglichst ideal ergänzen, um z.B. eine Doppelbelastung eines der Teammitglieder zu vermeiden und die zur Verfügung stehende Zeit optimal zu nutzen.

### 3 Anforderungen

Die Anforderungen an das Projekt wurden aus verschiedenen Quellen definiert. Hier zum Einen allein schon durch das Modulthema „**Web Technologie**“, sowie weiter im Allgemeinen sowie im Speziellen durch den **Dozenten** als auch abschließend auch durch uns als **Projektteam** selbst.

Das Modulthema gibt hierbei das grundsätzliche Umfeld vor und bestimmt damit auch wesentlich alle weiteren Details folgender Anforderungen. Der Dozent gab hier allgemeine Anforderungen an alle Teilnehmenden des Moduls, und definierte weiter auch spezielle Anforderungen an die Durchführung unseres Projektes. Eine dritte Menge von Anforderungen entstand aus unseren eigenen Projektbesprechungen und -abstimmungen.

All diese Anforderungen wurden gebündelt in der Projektbesprechung vom 27.11.2021 erfasst, von uns bewertet und zusammen in eine Projektskizze (siehe Abbildung ??) transportiert. Mit den darin erfassten Inhalten unter den Punkten **Zielsetzung, Aufgaben und Ergebnisse** sowie **Randbedingungen** wurden alle gestellten Anforderungen systematisch erfasst und damit sichergestellt, dass diese im Projektverlauf auch entsprechend Beachtung finden und erledigt werden.

Die wentsentlichen Anforderungen, deren Erfüllbarkeit vor allem auch an die gewählten Technologien gebunden sind, wurden wie folgt bestimmt: Das Spiel muss...

- ...im Web laufen.
- ...mehrspielerfähig sein.
- ...eine gewisse Komplexität (Charakterentwicklung, Klassen) haben.
- ...performant und stabil laufen.

Die Anforderungen „im Web laufen“ sowie „performant und stabil“ sind bei korrekter Konfiguration, entsprechender Hardware und Wartung durch nahezu jeden Stack<sup>1</sup> zu erfüllen.

Da das gesamte Projektteam unerfahren mit der Umsetzung von Webprojekten war und die ersten, in den Vorlesungen zum Modul gesammelten Erfahrungen, mit der dort vorgestellten Lösung **Django** durchweg positiv waren, wurde dem eine deutliche Präferenz zugeschrieben. Die zuerst genannten Anforderungen dürfen bei Django auch als erfüllt betrachtet werden. Django ist weit verbreitet und wird produktiv im gewerblichen Einsatz erfolgreich betrieben.

---

<sup>1</sup> Stack: Software-, Technologie- oder Lösungsstack

Die nächste Anforderung „mehrspielerfähig“, stellte bereits höhere Ansprüche an die eingesetzte Technologie. Zuerst war hier zu klären bzw. zu definieren, wie genau der Ablauf des Spiels sein soll. Auch musste festgelegt werden welche Arten von Interaktion zwischen dem Spiel und den Spielern und auch unter den Spielern selbst gewünscht sind.

Wir haben hier in Projektbesprechungen schnell erkannt, dass das Spiel mit allen möglichen Interaktionen, jeweils einzeln genau definiert werden muss. Nur so lassen sich konkrete Anforderungen formulieren und die spätere Programmierung erledigen.

Bei dieser Definition wurde weiter deutlich, dass es (bedingt durch die geforderte Mehrspielerfähigkeit) nicht nur einen bedeutenden Aufwand zu Entwurf und Programmierung bezüglich der Spiellogik selbst, sondern auch bezüglich des **Matchmaking**<sup>2</sup> geben wird.

Das Matchmaking sollte hier nicht automatisch erfolgen, sondern bewusst durch die Spieler. Ein gewisser Austausch ist dafür unabdingbar: Es wurde eine Chat-Funktion benötigt. Über eine Recherche dazu, wurden wir auf das **Websocket-Protokoll** aufmerksam. Gemäß der Beschreibung nach heise online, Matthias Wessendorf (2011) sollten unsere Anforderungen umfassend erfüllt werden. Das Websocket-Protokoll ist, ähnlich einer TCP-Verbindung, eine bestehende Verbindung, die es auch erlaubt Events von Serverseite aus an den Client (ohne dessen gesonderte Anforderung) zu übertragen.

An vielen Stellen im Projekt würde dies eine Anforderung sein, die mit den Websockets erfüllt werden konnten.

Alternative Frameworks (Backend, als auch Frontend) waren uns zwar teilweise vom Namen bekannt, wurden aber weiter nicht genauer in Betracht gezogen. Auch damit grundlegende Erfahrungen im Umgang mit HTML, CSS und JavaScript im Rahmen des Projekts durch das Team gesammelt werden können. Die Abkürzung über ein Framework wurde daher hier auch bewusst nicht gewählt.

---

<sup>2</sup> Das Matchmaking umfasst bei Mehrspielerspielen das Zusammenfinden oder auch automatische Zusammenstellen von verschiedenen Mitspielenden zu einem Spiel

## 4 Herangehensweise

Bereits in der frühen Konzeptphase des Projektes entschieden wir unsere Arbeit mithilfe des Projektmanagement-Modells nach Scheurer zu organisieren (Scheurer, Bea und Hesselmann (2020)). Da dieses umfangreiche Theorie, Werkzeuge und Best-Practices für Großprojekte bereitstellt, entschieden wir weiterhin nur die für unser vergleichsweise kleines Softwareprojekt sinnvollen Teile zu implementieren. In ersten informellen Brainstormings zu grundlegenden Themen wie Spieldesign, Gameplay-Mechaniken, Setting und Technik wurden die groben Rahmenbedingungen der Entwicklung abgesteckt. Diese wurden dann in einer „Kick-Off“-Veranstaltung konkretisiert und in einer detaillierten Projektsizze festgehalten (zu sehen in den folgenden Abbildungen).

**Abbildung 1: Projektsizze vom 27.11.2021**

(a)	(b)
<p><b>Projektskizze</b></p> <p>Samstag, 27. November 2021 09:28</p> <p><b>Projektskizze</b></p> <p><b>Name des Projektes:</b> &lt;Hier Spieldatei einsetzen&gt;</p> <p><b>Zielsetzung:</b></p> <ul style="list-style-type: none"> <li>• Multiplayer-Komponente</li> <li>• Meilensteine mit Terminen setzen (und einhalten :))</li> <li>• Benutzerverwaltung (Login etc.)</li> <li>• Charaktererstellung (Creator)</li> <li>• Balancing (Charakterfortschritt)</li> <li>• Weltkarte als Hub -&gt; Lobby -&gt; Spielszene</li> <li>• Weltkarte <ul style="list-style-type: none"> <li>◦ Statisches Hintergrundbild (+ Sprites?)</li> <li>◦ Einzelne Sprites / Grafiken als Szeneneinstiegspunkte</li> <li>◦ Anzeige der aktiven Spieler</li> <li>◦ Weltnachrichten</li> </ul> </li> <li>• Lobby <ul style="list-style-type: none"> <li>◦ Jede Szene hat eine Lobby</li> <li>◦ Füllstandsanzeige auf Weltkarte</li> </ul> </li> <li>• Spielszene <ul style="list-style-type: none"> <li>◦ Fortschrittsmechanik (mehrere aufeinander aufbauende "Level")</li> <li>◦ Kampfmechanik (Gegnerklassen etc.) / Ressourcenmechanik (Leben, Mana etc.)</li> <li>◦ Szenarten: Rätsel, Kampf, Charakterentwicklung</li> <li>◦ Events (Skriptereignisse)</li> <li>◦ Gestaltung im Stil eines klassischen RPG's (Szenenbild, Menüleiste, Kampfflog)</li> </ul> </li> <li>• Szeneneditor + Speicherung auf Datenbank</li> <li>• Musik / Ton</li> <li>• Datenbankanbindung</li> <li>• Spielverwaltung (Überwachung eines laufenden Spiels, Schließen von Szenen etc.)</li> <li>• Spiel muss durchspielbar sein</li> <li>• Projektarbeit schreiben</li> </ul>	<p><b>Aufgaben &amp; Ergebnisse:</b></p> <ul style="list-style-type: none"> <li>• Technische Machbarkeit sicherstellen</li> <li>• Technisches Grundgerüst bauen</li> <li>• Inhaltliches Konzept erstellen <ul style="list-style-type: none"> <li>◦ Setting</li> <li>◦ Grafikstil / Leveldesign</li> <li>◦ Musikstil</li> <li>◦ Spielfortschritt</li> <li>◦ Charaktere</li> <li>◦ GameLoop (Anfang-&gt;Mitte-&gt;Ende, Game Over Mechanik etc.)</li> <li>◦ Story</li> </ul> </li> <li>• Dokumentation (in Hinblick auf schriftlichen Teil der Arbeit)</li> </ul> <p>--&gt; Projektarbeit abgeben (Schrift + Code + Link)</p> <p><b>Risiken:</b></p> <ul style="list-style-type: none"> <li>• Technische Umsetzbarkeit</li> <li>• Fehlende Zeit</li> <li>• Fehlendes Know-How</li> <li>• Rechte (Grafiken / Musik / Code)</li> <li>• Personalmangel (Ausfall durch Krankheit / Jobs etc.)</li> <li>• Verstrickung in Kleinigkeiten / Fokusverlust</li> </ul> <p><b>Randbedingungen:</b></p> <ul style="list-style-type: none"> <li>• Spiel muss im Web laufen</li> <li>• Spiel muss Multiplayer-fähig sein</li> <li>• Spiel sollte eine gewisse Komplexität haben (nicht zu simpel)</li> <li>• Spiel muss performant und stabil laufen</li> </ul> <p><b>Termine:</b></p> <ul style="list-style-type: none"> <li>• 13.02.2022, 23:59 !!!</li> </ul> <p><b>Auftraggeber:</b></p> <ul style="list-style-type: none"> <li>• Daniel Bitzer / FOM</li> </ul> <p><b>Aufwand in Personentagen:</b></p> <p>&lt;Hier Meilensteine einsetzen&gt;</p>

Ziel einer solchen Skizze ist es, einen fokussierten Überblick über alle Aspekte des durchzuführenden Projektes zu gewinnen. Neben allgemeinen Informationen wie Projektname und Auftraggeber, werden Zielsetzung, Aufgaben, erwartete Ergebnisse, Risiken und Randbedingungen definiert. Auch werden wichtige Termine hervorgehoben und Meilensteine für einzelne Teiltätigkeiten gesetzt. Der Definition einer konkreten Zielsetzung kommt hierbei eine herausragende Bedeutung zu, da sich viele der nachfolgenden Aspekte aus eben dieser ergeben.

Den ersten Meilenstein des Projektes stellte die Entwicklung eines rudimentären Prototypen dar, der die technische und konzeptionelle Machbarkeit des Spieles demonstrieren

sollte. Darauf aufbauend konnte dann eine (grobe) Aufwandsschätzung erfolgen und es wurden weitere Meilensteine gesetzt.

In Anbetracht unserer eigenen Zielvorstellungen und der im obigen Kapitel erläuterten globalen Anforderungen, entschieden wir das Projekt in drei nebenläufige Teilprojekte zu unterteilen:

- Infrastruktur und Backend-Entwicklung (Henning Beier)
- Frontend-Entwicklung und Grafik-Design (Julian Schäfer)
- Game-Design (Rico Pursche)

Im Rahmen der „Infrastruktur- und Backend-Entwicklung“ sollte das technische Grundgerüst des Spiels erstellt werden. In diesen Bereich fielen Aufgaben wie die Bereitstellung der Server-Architektur, die Definition und Verwaltung der Datenbank und die Programmierung der Game-Logik.

Im Teilprojekt „Frontend-Entwicklung und Grafikdesign“ lagen die Gestaltung der Website mit HTML, CSS und JavaScript im Fokus. Ein weitere Aufgabe war die Erstellung sämtlicher Grafik-Assets, d.h. Concept Art, Bilder, Sprites, Icons und Animationen.

Setting, Gameplay-Konzept, das Schreiben von Texten, die Ausarbeitung des Kampfsystems und das damit verbundene Balancing lagen im Aufgabenbereich des Teilprojektes „Game-Design“.

Diese Aufteilung ist hierbei nicht als strikte Trennung zu verstehen, bei der die einzelnen Arbeitsbereiche voneinander abgeschottet sind. Vielmehr wurden Verantwortungsbereiche abgesteckt, die sich auch überschneiden können und das in der Umsetzung auch taten. So wurde häufig in „teilprojektfremden“ Tätigkeitsbereichen gearbeitet, was aber von Beginn an so vorgesehen war. Der Austausch von Daten, Informationen und die Versionsverwaltung wurde über ein gemeinsames Git-Repository realisiert. Änderungen wurden lokal getestet und dann in das Repository geschoben.

Zudem wurde ein regelmäßiges Meeting am Sonntag eingeplant. Hier wurden der allgemeine Projektstatus besprochen, die Einhaltung der gesetzten Meilensteine überprüft und gemeinsame Entscheidungen über verschiedenste Aspekte des Projektes getroffen. Ebenfalls wurden Konzepte für Teilthemen wie z.B. Frontend-Layout, Setting und viele Weitere erarbeitet und aufeinander abgestimmt (in den folgenden Abbildungen beispielhaft zu sehen).

**Abbildung 2: Frühes UI-Layout Mockup**



**Abbildung 3: Konzept: Kampfsystem vom 29.11.2021**

**Kampfblauf:**

- Begrüßungstext ausgeben ("Hoho, Ihr bekommt Prinzessin Peach nie!")
- Kampf läuft = true
- While(Kampf läuft):
  - Gegner fügt Schaden zu
  - für jeden (lebendigen) Spieler einzeln (in Reihenfolge der Slots):
    - Auswahl Aktion bestimmen (immer 3 Optionen: Schaden machen, Fähigkeit nutzen, Aussetzen [Aggro abbauen])
      - Aktion für Runde speichern
      - Aktionen durchgeführt (div. Rechnungen vorgenommen)
      - Gamelog wird fortgeschrieben
    - Aggrotabelle fortschreiben (Schaden 1:1 Aggro, Verspotten +100 Aggro)
    - stetige Prüfung ob ein HP unter 0 fällt:
      - Wenn Gegner unter 0: Abbruch: Win
      - Wenn kein Spieler mehr am leben ist: Game Over

	HP	DMG	Fähigkeit
Gegner:	1500	80-120	
Krieger:	500	20-40	Verspotten (Aggro aufbauen, klingt über Runden ab)
Zauberer:	200	60-80	Buffen (Schaden der Gruppe für x Runden erhöhen)
Priester:	300	30-60	Heilen (HP der Gruppe regenerieren [auch über Runden?])

Levelfortschritt nach Abschluss eines Levels:

- Gegner erledigt?
  - ja: HP + 5-10 %, DMG min +8%, DMG max +10%
  - nein: nix (kein Game-Progress)

Backlog:

- Mehr Klassen mit anderen Fähigkeiten (Totstellen, Verbllassen, ...)

## 5 Vorstellung der Ergebnisse

In diesem Kapitel soll das von uns entwickelte Spiel „Monster Slayer“ vorgestellt werden. Dazu wird zunächst ein Überblick über Konzept und Spielablauf anhand des unten zu sehenden Ablaufdiagramms 4 gegeben. Darauf folgen nähere Erläuterungen zu verschiedenen Teilespekten des Spiels, wie Game-Design, Grafik-Design, Frontend und Backend.

In „Monster Slayer“ übernimmt der Spieler die Rolle eines Söldners, der durch das Land zieht und sein Geld als Monsterjäger verdient (ähnlich dem „Witcher“ aus den Romanen von Andrzej Sapkowski). Das Spiel besitzt sowohl Einzelspieler- als auch Mehrspielerkomponenten. Letztere sind auf eine Art und Weise konzipiert, die ein erfolgreiches Bestreiten der Inhalte nur durch Zusammenarbeit mit anderen Spielern möglich macht. Kommunikation und Kooperation mit den Mitspielern sind hier erwünscht und notwendig.

Nach der Anmeldung auf der Website erhält der Spieler die Möglichkeit sich einen oder mehrere Charaktere zu erstellen. Diesen Charakteren kann eine von drei verfügbaren Klassen zugewiesen werden: Krieger, Priester und Zauberer. Diese Klassen besitzen die Attribute Lebenspunkte (HP) und Angriffskraft (AP), welchen je nach Klasse feste Startwerte zugewiesen sind. Mithilfe von Erfahrungspunkten (XP) können diese Attribute während des Spielverlaufs gesteigert werden. Außerdem hat jede Klasse eine Spezialfähigkeit, die im Kampf eingesetzt werden kann. Das taktische Einsetzen dieser Fähigkeiten kann im Spiel über Sieg oder Niederlage entscheiden.

Die Klassen unterscheiden sich wie folgt:

Klasse	Rolle	Statuswerte	Fähigkeit
Krieger	Tank	200 HP, 40 AP	Verringert Angriffspunkte des Gegners
Priester	Heiler	180 HP, 30 AP	Stellt die Lebenspunkte von sich und der Gruppe wieder her
Zauberer	Schaden	160 HP, 50 AP	Erhöht die Angriffspunkte der Gruppe

Nach Abschluss der Charaktererstellung kann der Spieler die Weltkarte betreten. Auf der Karte sind eine Reihe von Orten hervorgehoben, die verschiedene Level (im Kontext der Entwicklung Szenen genannt) darstellen und von dem Spieler betreten werden können. Innerhalb dieser Level ist es die Aufgabe des Spielers/der Spieler einen Gegner im Kampf zu bezwingen. Es gibt drei Einzelspieler- und zwei Mehrspieler-Level. Die drei Einzelspieler-Level besitzen einen aufsteigenden Schwierigkeitsgrad. Schließt der

Spieler alle drei erfolgreich ab, ist das Spiel zum jetzigen Entwicklungsstand „durchgespielt“. Die Mehrspieler-Level sind für den Spielfortschritt optional und können als Endgame-Content verstanden werden. Hier hat der Spieler auf der Weltkarte die Möglichkeit mithilfe eines globalen Chats Mitstreiter zu finden. Alle Level sind zudem beliebig oft wiederholbar.

Es gibt folgende Level:

- Einzelspieler
  - Level 1: Am Rande des Dunkelwaldes; Gegner: Warg
  - Level 2: Das Alte Anwesen; Gegner: Vampirfürst
  - Level 3: Der Ascheberg; Gegner: Drache
- Mehrspieler
  - Level 4: Der Friedhof in den Sümpfen, Gegner: Untoter Magier
  - Level 5: Die verlassene Mine; Gegner: Riesenwurm

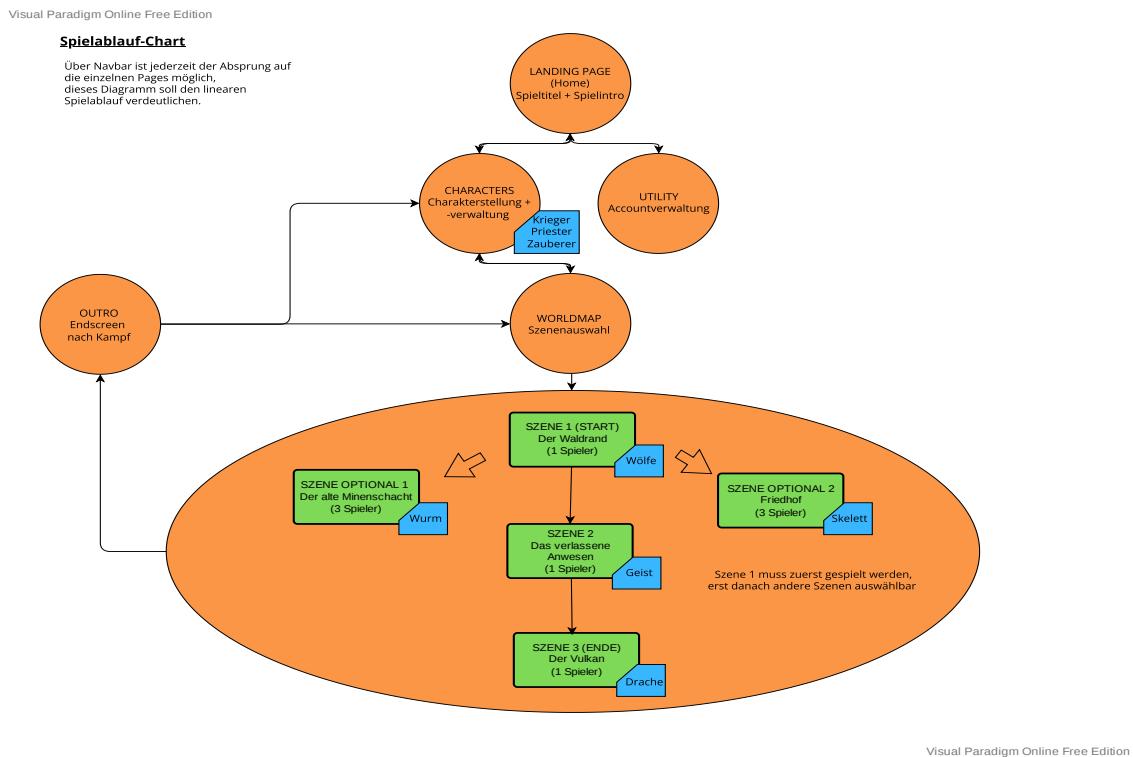
Wenn der Spieler sich für ein Level entschieden hat, wird er in eine Lobby weitergeleitet. Auch hier kann über ein Chatfenster mit anderen Spielern in der Lobby kommuniziert werden. Jedes Level hat eine feste Anzahl von Spielerslots, die mit Klick auf einen entsprechenden Button „belegt“ werden können. Sind alle Spielerslots belegt, startet ein Countdown und der Spieler kann das Level betreten.

Im Level angekommen steht dem Spieler ein Bossgegner gegenüber, den es zu besiegen gilt. Der Kampfbildschirm ist aufgeteilt in eine Grafik des Gegners mit einem animierten Gegner und Hintergrund, einen Status- und Aktionsbereich für den Spieler, ein Gamelog und ein Inputfeld für Chatnachrichten (welche im Gamelog angezeigt werden). Kämpfe laufen rundenbasiert ab, wobei der Gegner immer zuerst an der Reihe ist. Hat dieser seine Aktion ausgeführt, wartet das Spiel auf die Aktion des Spielers. Dieser hat nun die Möglichkeit entweder einen Angriff auszuführen oder seine Klassenfähigkeit einzusetzen. Ein Aussetzen ist ebenfalls möglich. Handelt es sich um einen Mehrspieler-Kampf, wartet das Spiel auf die Eingabe aller beteiligten Spieler, bevor die Runde voranschreitet. Im Gamelog wird kenntlich gemacht, wenn Spieler ihre Aktionen ausgewählt haben und auf die Eingabe der anderen warten. Die Stärke des Angriffs von Spielern und Gegner ergibt sich aus den Angriffspunkten des jeweiligen Charakters und einer darauf basierenden Zufallskomponente.

Der Kampf endet, wenn die Lebenspunkte des Bossgegners oder die aller beteiligten Spieler auf null fallen. Der erste Fall resultiert in einem Sieg für die Spieler, die nun

Erfahrungspunkte plus einen Siegbonus erhalten. Der zweite Fall resultiert in einer Niederlage, die Spieler erhalten aber trotzdem Erfahrungspunkte. In beiden Fällen werden die Spieler zu einem Spielabschluss-Bildschirm weitergeleitet von wo aus die Rückkehr zur Weltkarte oder in die Charakterauswahl möglich ist. Die erhaltenen Erfahrungspunkte können jetzt genutzt werden, um die Charakterattribute zu verbessern und auf der Weltkarte können weitere Level ausgewählt werden.

**Abbildung 4: Diagramm: Seitenaufbau und Spielablauf**



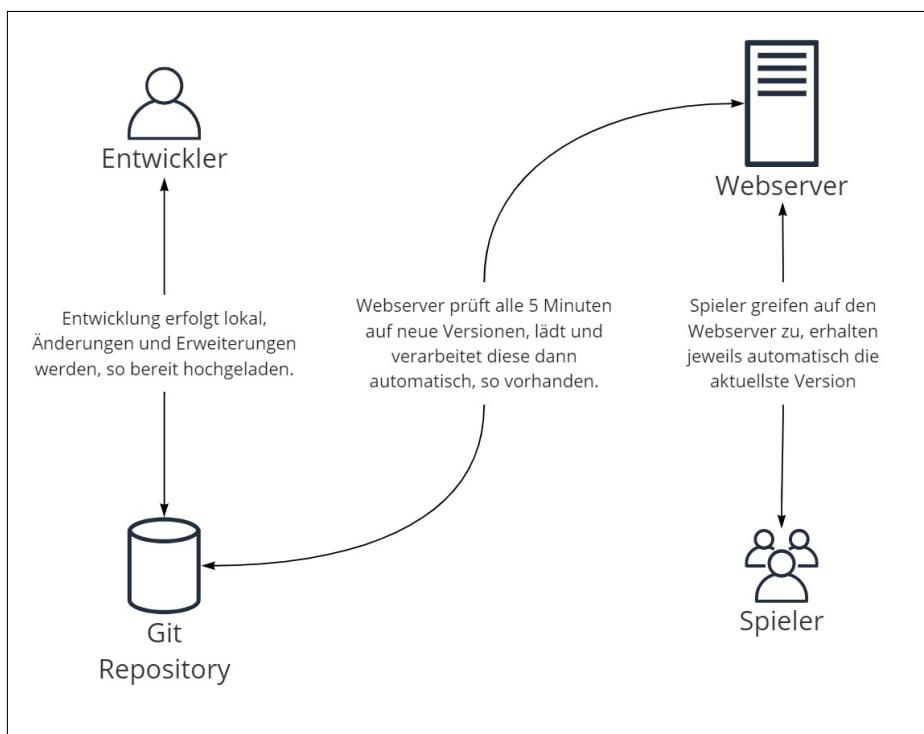
Abschließend sei erwähnt, dass sich das Spiel in gewisser Weise in einem „Beta“-Status befindet. Alle oben beschriebenen Inhalte wurden planmäßig umgesetzt und funktionieren grundsätzlich auch. Der „Veröffentlichung“ des Spiels ist allerdings keine eingehende Testphase vorausgegangen, etwaige Spielfehler können also nicht gänzlich ausgeschlossen werden. Refactoring wurde auch nur sehr rudimentär betrieben, da der Programmcode bis kurz vor „Veröffentlichung“ noch häufigen Änderungen unterworfen war.

## 5.1 Infrastruktur und Code-Entwicklung

### 5.1.1 Infrastruktur

Die für dieses Projekt eingesetzte Infrastruktur setzt auf verschiedene Komponenten, deren Zusammenspiel hier in einer vereinfachten Darstellung zu sehen ist und im Folgenden näher beschrieben wird:

**Abbildung 5: Konzept: Entwicklung, Codeverteilung, Produktivnahme**



Die Dienste selbst laufen auf einem für dieses Projekt bei der **HostEurope GmbH<sup>3</sup>** angemieteten, virtuellen Server. Das Produkt nennt sich „Virtual Server 10.0“ und verfügt in der geringsten Ausstattung über 1 vCPU, 2 GB RAM sowie 40 GB SSD-Speicher. Auf dem Server läuft Debian 11. Der Server hat außerdem eine feste IP-Adresse. Einrichtungskosten gab es keine. Der Vertrag ist monatlich kündbar. Der Mietpreis beträgt 5,99 EUR incl. MwSt. monatlich.

Weiter wurde bei **INWX GmbH<sup>4</sup>** zu dem Server auch eine Domain gekauft sowie entsprechende Namenservereinträge hinterlegt, die den Betrieb eines Traefik-Proxys mit Subdomains zu Domain erlauben. Auch hier gab es keine Einrichtungskosten. Der Preis für die Domain beträgt pro 5,97 EUR incl. MwSt. pro Jahr.

<sup>3</sup> Siehe <https://www.hosteurope.de/Server/Virtual-Server/>.

<sup>4</sup> Siehe <https://www.inwx.de/de/de-domain>.

Abseits der vorgenannten Hardware, stellt das gemeinsame **Git-Repository auf GitHub**<sup>5</sup> einen zentralen Punkt der Infrastruktur dar. In diesem Repository liegen alle Elemente des Projektes: Code, Skripte und Dokumentation.

Die Funktionen des Repositorys sind folgende:

- Entwickler übermitteln neue Inhalte in das Repository (push).
- Entwickler erhalten den stets aktuellen Entwicklungsstand aus dem Repository (pull, fetch).
- Der Server lädt neue Inhalte aus dem Repository automatisch und aktualisiert die Dienste.
- Alle Beteiligten und auch Dritte können über das Repository die Entwicklung nachvollziehen.

Für den Update-Prozess des Servers und der Dienste wurde ein Cron-Skript erstellt, dass laufend prüft ob neue Commits im Repository vorhanden sind (siehe **Codelisting 1** unten, Zeile 5). Wenn ja, wird der Django-Webserver gestoppt (Zeile 7), die neuen Inhalte heruntergeladen, verarbeitet und der Django-Webserver im Anschluss wieder gestartet (Zeile 15).

#### **Codelisting 1: „rpg/server-update-cron.sh“, gekürzt um Logging:**

```
1 #!/bin/bash
2 cd /home/rjadmin/tstsrv/
3 now=$(date "+%F %H:%M:%S")
4 git -C /home/rjadmin/tstsrv/ fetch origin
5 if [ `git -C /home/rjadmin/tstsrv/ rev-list HEAD...origin/main --count` != 0 ]
6 then
7     docker-compose --project-directory /home/rjadmin/tstsrv/ stop rpg
8     git -C /home/rjadmin/tstsrv/ reset --hard origin/main
9     git -C /home/rjadmin/tstsrv/ fetch
10    git -C /home/rjadmin/tstsrv/ pull
11    chmod +x /home/rjadmin/tstsrv/*.sh
12    docker-compose --project-directory /home/rjadmin/tstsrv/ run rpg python rpg/
13        manage.py makemigrations
14    docker-compose --project-directory /home/rjadmin/tstsrv/ run rpg python rpg/
15        manage.py migrate
16    docker-compose --project-directory /home/rjadmin/tstsrv/ run rpg python rpg/
17        manage.py loaddata db_sample_data.json
18    docker-compose --project-directory /home/rjadmin/tstsrv/ start rpg
```

Ein weiterer wesentlicher Punkt der Software-Infrastruktur sind die **Docker-Container** für den Django-Server, die Datenbank und den Reverse-Proxy. Hier stehen für die lokale Entwicklung „docker-compose“-Skripte für Windows und auch unixartige Systeme

---

<sup>5</sup> Siehe <https://github.com/tstsrv-de/rpg/>.

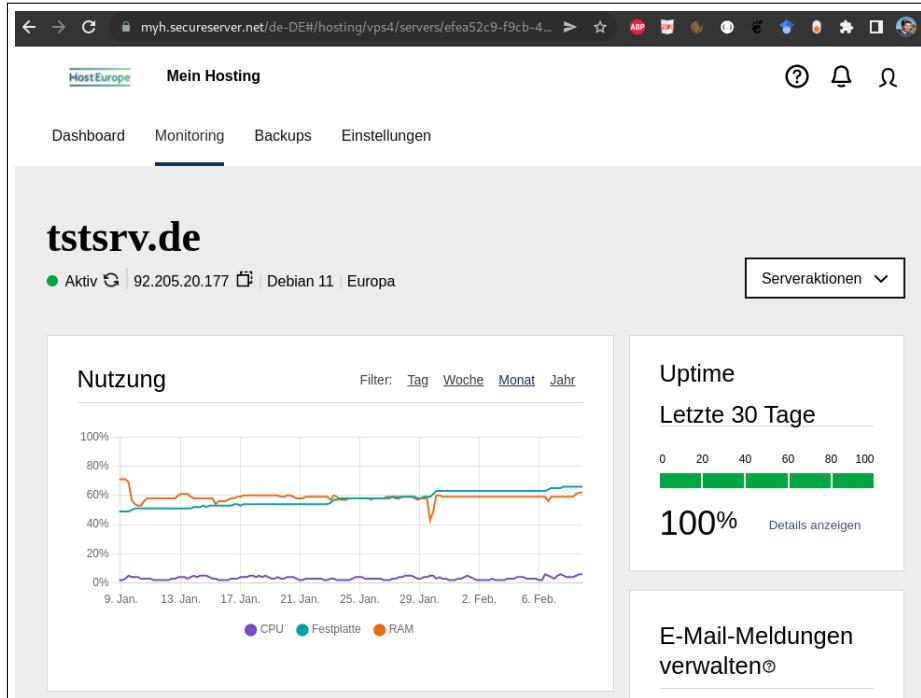
wie Linux und MacOS zur Verfügung. Ebenso für den Betrieb des Servers. Details zur Einrichtung und Nutzung wurden in der **Readme Repositorys<sup>6</sup>** hinterlegt.

Der Traefik-Proxy wurde so eingerichtet, dass Anfragen auf den Ports HTTP 80 und HTTPS 443 angenommen werden. Anfragen an HTTP 80, werden automatisch umgeleitet an HTTPS 443. Die notwendigen Zertifikate für die Subdomains werden vollautomatisch vom Traefik-Proxy über *Letsencrypt* angefordert und verwaltet. Als Grundlage für die Konfiguration diente eine Vorlage von Igor Bubelov<sup>7</sup> die entsprechend um die Funktionen für den Django-Webserver und die Datenbank erweitert wurden.

Die hier entwickelte Konfiguration erwies sich als so verlässlich, dass auch ein anderes Projektteam (**Deskshare<sup>8</sup>**) aus unserem Studiengang, ihre Dienste auf unserer Hardware unter einer eigenen Subdomain betreiben konnte. Die Anbindung an den Traefik-Proxy war möglich, auch die SSL-Zertifikate wurden vollautomatisch erstellt. Weiter konnte auch das Update-Konzepts mittels Cron-Dienst und Github-Repository übernommen werden, so dass das andere Projektteam eigenständig entwickeln konnte.

Die Ressourcen des Servers reichen hier für beide Projekte, auch in der Phase vor Abgabe der Arbeiten, aus:

**Abbildung 6: Bildschirmfoto: Server Auslastung**



<sup>6</sup> Siehe <https://github.com/tstsrv-de/rpg/blob/main/README.md>.

<sup>7</sup> Siehe Github-Repository dazu <https://github.com/bubelov/traefik-letsencrypt-compose>.

<sup>8</sup> Siehe <https://github.com/tstsrv-de/deskshare> bzw. <https://deskshare.tstsrv.de/>.

### 5.1.2 Initialisierung der Django-Anwendung

Nach dem Aufbau der grundsätzlichen Funktionen des Servers (Update-Systemdienste und Cron-Skripte), wurde die Django-Anwendung initialisiert. Der Ablauf orientierte sich hier an den in den Vorlesungen gewonnenen Erkenntnissen und wurde im Anhang (siehe Anhang Anhang 3) stichpunktartig festgehalten.

Im Anschluss daran wurden dann zuerst grundlegende Funktionen für die Benutzer-Regeistrierung am System hinzugefügt. Das geschah insbesondere unter Nutzung von zwei Anleitungen<sup>9</sup>.

Da das Projekt selbst die Entwicklung eines Spiels umfasste, wurde hier bewusst auf die Implementation von allen ansonsten mit einer Benutzerverwaltung im Zusammenhang stehenden Funktionen verzichtet. Es ist somit **nicht möglich**, seinen Benutzer z.B. zu löschen oder **ein vergessenes Passwort** zu erneuern. Unser Fokus lag hier, den Anforderungen entsprechend, auf der Entwicklung der Hauptanwendung.

Es wird hier in dieser Dokumentation auch nicht weiter auf die implementierte Benutzerverwaltung eingegangen, da es sich um weithin bekannte Standart-Features von Django handelt.

Mit Abschluss der Einrichtung der Django-Anwendung, liegt ein mit den bekannten Funktionen (wie Datenbankmodelle, Benutzerverwaltung, DB-Migrationen, ...) nutzbarer Django-Webserver vor. Für die Entwickler und die Entwicklung selbst ist die direkte Interaktion mit dem Django-Webserver nicht mehr notwendig. Die entwickelten Skripte starten den Webserver mit allen notwendigen Parametern selbstständig und führen vorher auch notwendige Schritte wie „makemigrations, migrate oder loaddata“ automatisch aus. Beispielhaft hier ein Auszug aus dem Skript für den Start der lokalen Entwicklungs-Umgebung unter unixartigen Betriebssystemen<sup>10</sup>:

```
1 #!/bin/bash
2 docker-compose -f local-dev-docker-compose.yml up -d
3 docker-compose -f local-dev-docker-compose.yml exec rpg python rpg/manage.py
   makemigrations
4 docker-compose -f local-dev-docker-compose.yml exec rpg python rpg/manage.py
   migrate
5 docker-compose -f local-dev-docker-compose.yml exec rpg python rpg/manage.py
   loaddata db_sample_data.json
6 docker-compose -f local-dev-docker-compose.yml stop
7 docker-compose -f local-dev-docker-compose.yml up
```

---

<sup>9</sup> Siehe <https://www.nintyzeros.com/2020/06/login-register-user%20page-in%20django.html> und <https://docs.djangoproject.com/en/3.2/topics/auth/default/#built-in-auth-forms>.

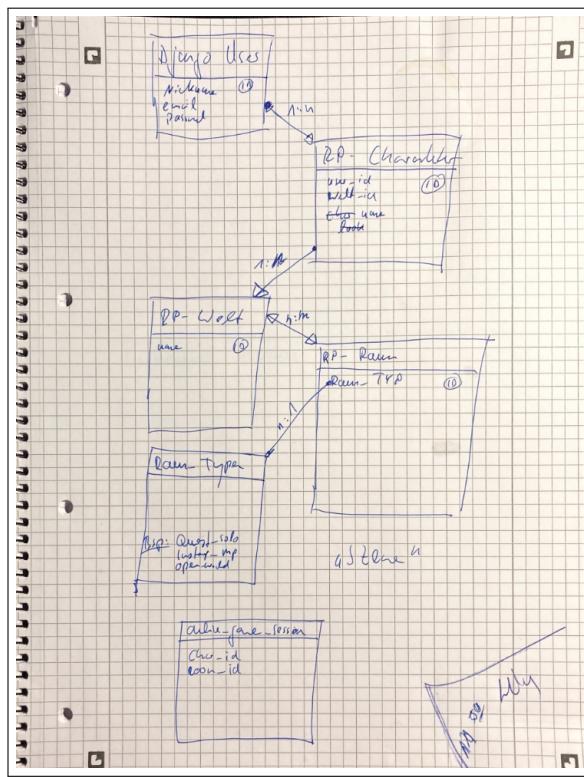
<sup>10</sup> Siehe <https://github.com/tstsrv-de/rpg/blob/main/local-dev-start.sh>.

### 5.1.3 Datenbankmodell

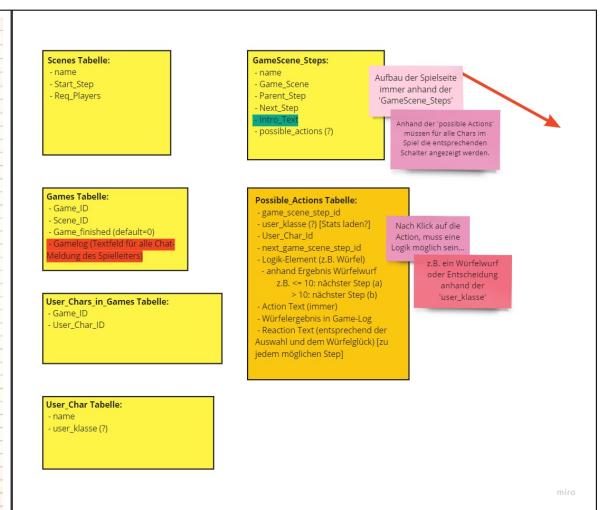
Aus den ersten Gesprächen und Austausch zum Projekt und den später formulierten Anforderungen wurden erste Konzepte und Ideen zu Papier gebracht. Dabei wurden auch erste Zeichnungen zu einem möglichen Datenbankmodell erstellt. In bzw. zu den späteren Projektbesprechungen wurden konkretere Zeichnungen erstellt.

**Abbildung 7: Datenbankmodell Entwurfsprozess**

(a) Entwurfszeichnung vom 23.11.2021



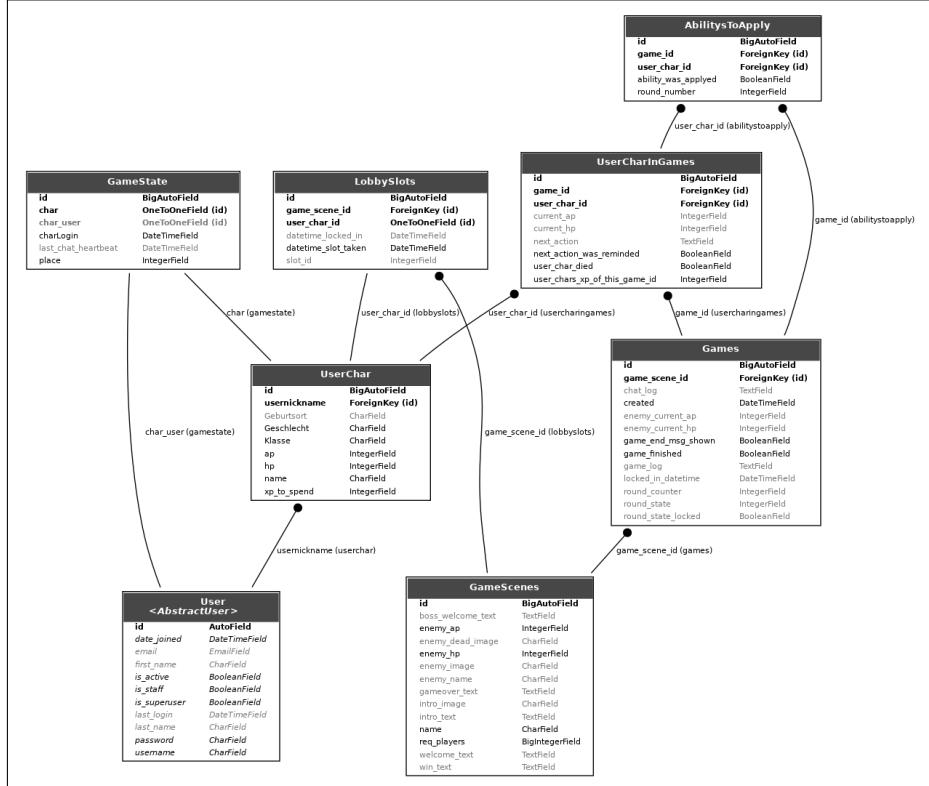
(b) Weiterentwickelter Entwurf (05.12.2021)



Fortsetzung auf der Folgeseite.

Hier eine Darstellung des Datenbankmodells als Diagramm sowie anschließend die ausführliche Beschreibung jeweils mit Stand vom 10.02.2022.

**Abbildung 8: Datenbankmodell final (Stand 10.02.2022)**



Der Aufbau der des Datenbankmodells beginnt logisch mit der **Benutzertabelle**, sowie den anderen von Django systemseitig erstellten Tabellen, auf die hier nicht weiter eingegangen wird.

Auf die Benutzertabelle setzt eine **Tabelle für Charaktere** (*UserChar*) auf. Darin enthalten und jeweils einem Benutzer zugeordnet, sind alle Daten zu den Spielercharakteren. Neben dem Namen und der Klasse sind in der Tabelle auch veränderliche Attribute wie die Angriffspunkte und Lebenspunkte sowie noch zu verteilende Erfahrungspunkte abgespeichert.

Bei der Erstellung eines neuen Charakters werden die Start- bzw. Standartwerte für Lebens- und Angriffspunkte aus einer besonderen **Konfigurationstabelle** (*myrpgconfig*) abgerufen und für die Erstellung des neuen Charakters, entsprechend der ausgewählten Klasse, abgespeichert. In diese Konfigurationstabelle wurden weiter auch **alle Konstanten**, die im Laufe der Entwicklung Einzug in den Programmcode gehalten haben transportiert. Dies umfasst tatsächlich komplett alle Konstanten. Beispielhaft genannt

seien hier z.B. der Faktor für die Berechnung von Erfahrungspunkten aus dem verursachten Schaden, der Minimal- und Maximalfaktor bei der Berechnung des bei einem Angriff entstandenen Schaden aus den Angriffspunkten heraus sowie auch die Laufzeiten in Rundenzahl sowie die Stärke der Fähigkeiten der einzelnen Klassen.

Das erlaubt es die gesamte Konfiguration des Projektes und ebenso auch das Balancing über die Administrationsoberfläche von Django unkompliziert zu verändern und zu pflegen. Die Konfigurationstabelle umfasst 21 Einstellungen. Zu jeder Einstellung sind in der Datenbank selbst auch entsprechende Erklärungen zur genauen Verwendung und Nutzung hinterlegt. Auch ist es möglich **verschiedene Datentypen in der Konfigurationstabelle** abzuspeichern und abzurufen. Folgende Hilfsfunktion wurde dazu in der Datei „rpg\_tools.py“<sup>11</sup> erstellt:

```
1 def rpg_get_config(config_to_get):
2
3     try:
4         config_type = MyRpgConfig.objects.get(name=config_to_get).type
5
6         if config_type == "int":
7             return int(MyRpgConfig.objects.get(name=config_to_get).value)
8
9         elif config_type == "str":
10            return str(MyRpgConfig.objects.get(name=config_to_get).value)
11
12        elif config_type == "float":
13            return float(MyRpgConfig.objects.get(name=config_to_get).value)
14
15    (...)
```

Neben der Benutzer-, Charakter- und der Konfigurationstabelle gibt es eine ganze Menge an Tabellen für das Spiel selbst. Zentral ist dabei die **Grundtabelle der Level (Game-Scenes)**: Diese ist als Bauplan der Level zu verstehen. Enthalten sind alle notwendigen Informationen um ein neues Spiel in einem der Level starten zu können: Der Name des Gegners, die Texte und Werte des Gegners. Aber auch die Anzahl der für diesen Level notwendigen Spieler. Ebenso sind alle Texte zum Level (Intro, Gewinn- und Gameover-Nachricht etc.) darin enthalten.

In der **Hilfstabelle für den Spieler-Staus (GameState)** wird jeweils der aktuelle Standort eines Spielers hinterlegt. Mit Standort ist dabei die aktuell aufgerufene Seite des Spieles gemeint. Es kann darüber nachvollzogen werden ob der Spieler sich gerade auf der Weltkarte befindet oder in einer der Level-Lobbys. Auch wird darin ein Zeitpunkt des letzten Datenpaketes aus dem Chat hinterlegt. Darüber ist es möglich, Spieler nach Ablauf einer gewissen Zeit (Sekunden), automatisch aus dem Chat zu entfernen.

---

<sup>11</sup> Siehe [https://github.com/tstsrv-de/rpg/blob/main/rpg/rjh\\_rpg/rpg\\_tools.py#L63-L80](https://github.com/tstsrv-de/rpg/blob/main/rpg/rjh_rpg/rpg_tools.py#L63-L80).

Innerhalb der Level-Lobby können über die **Hilfstabelle für die Level-Lobbys** (*LobbySlots*) die bereits belegten Plätze für ein Spiel nachgehalten werden. Die Spieler belegen in der Lobby mit einem Klick einen verfügbaren Platz. Sind dann alle Plätze des Levels belegt, wird nach Ablauf eines Countdowns **ein neues Spiel gestartet**.

Mit Spielstart wird in der **Spieltabelle** (*Games*) ein neues Spiel angelegt. Dieses Spiel darin trägt eine Verknüpfung zum Bauplan (Grundtabelle der Level: *GameScenes*), ein Game-Log in dem alle Texte des Spiels gespeichert werden, die aktuellen Lebens- und Angriffspunkte des Gegners sowie auch alle **Informationen zum Rundenstatus** des Spiels.

Gleichzeitig mit der Anlage des Spiels selbst, werden in der **Hilfstabelle für die Spieler- und Spiel-Zuordnung** (*UserCharInGames*) für jeden für das Spiel angemeldeten Spieler ein entsprechender Eintrag erzeugt. Während des Spiels werden darin alle Werte des Spieler hinterlegt. Auch ob der Spieler in diesem Spiel bereits verstorben ist, wird darin gespeichert.

Letzte mit dem Spielablauf im Zusammenhang stehende Tabelle ist eine **Spezialtabelle für die aktivierten Fähigkeiten der Spieler** (*AbilitysToApply*). Darin wird immer wenn in einem Spiel ein Spieler seine Fähigkeit aktiviert, alle Informationen abgespeichert die für die spätere Wirksamkeit notwendig sind. Entscheidet sich z.B. der Priester in Runde 3 seine Fähigkeit anzuwenden (und wirkt diese über 4 Runden), werden in der Tablle entsprechende Einträge für Runde 4, 5, 6 und 7 mit Bezug auf den Priester und die Heilung hinterlegt. Wendet die Runden- bzw. Spiellogik diese Fähigkeit an, wird ein entsprechndes Kennzeichen dazu in dem Datensatz gesetzt (*ability\_was\_applied = true*).

#### 5.1.4 Hervorzuhebende Code-Entwicklungen

Neben Standard-Entwicklungen wie z.B. Registrierungs- und Loginfunktionen (die hier wie bereits klar gemacht, nicht weiter beschrieben werden sollen), wurden vor allem sehr spezifische Entwicklungen durchgeführt. Im nun folgenden Abschnitt werden insbesondere zwei Teile näher beschrieben:

1. Die Funktionsweise und Implementation der Websockets, sowie
2. die Rundenlogik im Spielverlauf.

Zu den **Websockets** wurden im Kapitel Anforderungen einige Grundlagen bereits erklärt. Bei der Entwicklung wurde hier als erster Programmteil nach den Basisfunktionen, eine Chat-Funktion über die Websockets implementiert.

Dies vor allem unter Nutzung einer Anleitung<sup>12</sup>, die eine Basis-Form der Chat-Funktionalität bereitstellte. Diese wurde schnell für die in diesem Projekt beachtigten Zwecke erweitert und angepasst:

In der nun vorliegenden Lösung öffnet sich beim **Besuch der Worldmap**, über das dargestellte Chatfenster eine Websocket-Verbindung. Diese bleibt als offene Verbindung bestehen und erlaubt es Nachrichten zwischen dem Server (Django) und dem Client (Browser) auszutauschen. Auf der Serverseite läuft dies innerhalb Djangos über ein Routing der Websockets in entsprechende Consumer-Funktionen. Auf Clientseite findet JavaScript Verwendung für das Versenden sowie Empfangen und Verarbeiten der Nachrichten über das Websocket-Protokoll.

Innerhalb der **Level-Lobby-Seiten** wird jeweils auch eine entsprechende Websocket-Verbindung aufgebaut. Über die ID der Lobby, wird hier analog eine Websocket-Verbindung mit gleicher ID aufgebaut. Die Zuordnung des Aufrufs erfolgt hier innerhalb von Django über die in der *settings.py* hinzugefügte App *channels* sowie die Einstellungen zur ASGI-App **mit dem Routing-Ziel** zur entsprechenden Auflösung des Websockets-Aufrufs über die *routing.py* innerhalb der Django-App *rjh\_rpg*:

```
1 websocket_urlpatterns = [  
2     (...)  
3     re_path(r'ws/lobby-(?P<scene_id>\w+)/$', consumer_lobby.Consumer.as_asgi()),  
4     re_path(r'ws/game-(?P<game_id>\w+)/$', consumer_game.Consumer.as_asgi())  
5 ]
```

Der Aufbau von Websocket-Verbindungen ist, wie man sieht, analog der aus Django bekannten „urls.py“ geregelt. Von **Client-Seite** aus wird hier eine bestimmte URL aufgerufen und damit die Websocket-Verbindung aufgebaut. Über das Objekt dieser aktiven Verbindung können dann Nachrichten (hier als JSON-String) übermittelt werden, beispielhaft und stark vereinfacht wie folgt:

```
1 const chat_websocket = new WebSocket("wss://rpg.tstsrv.de/worldmap-chat/");  
2 chat_websocket.send(JSON.stringify({ user: user_id, message: message, }));
```

Als nächsten Entwicklungsschritt wurden hier dann die **veränderlichen Webseiten-inhalte über Websockets** transportiert - Ähnlich AJAX bzw. einem Framework mit Nutzung einer API als Backend. In dem Anwendungsfall unseres Spiels findet hier die Verarbeitung vor allem in dem Websockets-*Consumer* für das Spiel statt: der **consumer\_game.py**.

Darin werden vom Client erhaltene Nachrichten verarbeitet. Diese Nachrichten können z.B. das Klicken eines der Buttons sein (Angriff, Fähigkeit, Aussetzen) oder auch ein

---

<sup>12</sup> Siehe <https://github.com/veryacademy/YT-Django-Project-Chatroom-Getting-Started>.

laufend übersendetes Lebenszeichen, ein *Heartbeat*. Dieser teilt dem Server mit, dass der jeweilige Spieler noch im Spiel ist und die Webseite offen hat, die Websocket-Verbindung also noch Bestand hat. Stark vereinfacht und gekürzt stellt sich das im Programmcode wie folgt dar:

```

1 class Consumer(AsyncWebsocketConsumer):
2
3     # Verbindungsauftbau
4     async def connect(self):
5         self.game_id = self.scope['url_route']['kwargs']['game_id']
6         self.msg_group_name = 'game-%s' % self.game_id
7
8         await self.channel_layer.group_add(
9             self.msg_group_name,
10            self.channel_name
11        )
12
13     await self.accept()
14
15     # Neue Nachricht alle Empfänger der Websockets-Gruppe (hier Spieler des
16     # Spiels) senden
17     async def msg_group_send_game_log_update(self, event):
18         game_log_content = await db_get_game_log(self.game_id)
19
20         if game_log_content == self.last_game_log_content:
21             # skip sending if no change since last message
22             pass
23         else:
24             await self.send(text_data=json.dumps({
25                 'game_msg_type': 'game_log_update',
26                 'game_websocket_content': str(game_log_content),
27             }))
28             self.last_game_log_content = game_log_content
29
30     # Empfang und Verarbeitung erhaltener Nachrichten
31     async def receive(self, text_data):
32         text_data_json = json.loads(text_data)
33         message = text_data_json['msg']
34
35         if message == 'alive':
36             round_state = await db_get_round_state(self.game_id)
37
38             # Hier finden sich rund 300 Zeilen Code der die Spiel- und
39             # Rundenlogik abbildet
40
41             # Mit Ende der Verarbeitung werden neue Informationen an die Spieler
42             # des Spiels übersandt
43
44             await self.channel_layer.group_send(self.msg_group_name, {
45                 'type': 'msg_group_send_game_log_update', })

```

Man sieht, dass die Spielelogik im Wesentlichen durch den Erhalt von Nachrichten fortgeführt bzw. -geschrieben wird. Das Spiel durchläuft dabei Runden, die in Rundenstufen unterteilt sind. Innerhalb dieser Rundenstufen werden verschiedene Aktionen oder Prü-

---

funken durchgeführt. Auch werden in einer Rundenstufe z.B. die Aktionen der Spieler eingesammelt. Das bedeutet, dass das Spiel solange in dieser Rundenstufe (Aktionen einsammeln) verbleibt, bis jeder Spieler eine nächste Aktion ausgewählt hat.

In der Projektbesprechung vom 11.12.2021 wurde ein Rundenablauf erarbeitet, der übersetzt als Pseudo-Code Grundlage für die Programmierung wurde:

1. Aktion von Gegner ausführen (Schaden) [100]
2. Prüfen ob User-Char tod ist (HP < 1 = Dead-Flag: True) [200]
3. Prüfen wie viele User-Chars noch leben (n < 1 = Gameover-Flag: True, break-Gameloop-Schleife) [300]
4. Aktionen der User-Chars aufnehmen (Entscheidung für nächste Aktion von jedem Spieler annehmen + wegspeichern) [400]
5. Alle Aktionen der User-Chars ausführen (Aktionen laden und ausführen: Schaden, Aktion, Passen) [500]
6. Nach jedem Spieler, prüfen ob Gegner besiegt wurde (HP < 1 = Win-Flag: True, break-Gameloop-Schleife) [immernoch 500]
7. Rundencounter +1 [600]
8. Gameloop-Schleife nächster Durchlauf [700, zurück zu 100]...

Mit Abschluss der Programmierung ergab sich ein etwas detaillierterer Rundenablauf, der im Code dann wie folgt fixiert wurde und hier dokumentiert ist:

```

1 # all round states in game-loop:
2   0 initial for new games
3   50 apply effects of used abilities
4   100 enemy makes damage
5   200 check user-chars with less than 0 hp, mark them as dead
6   300 check if any user-chars are alive, if not end game with gameover-flag 990
7   400 collect and save the next actions from all alive user-chars, proceed only
        every user-char has a next action set
8   500 run the collected actions (make damage, use ability, pass turn), check after
        each user-char if enemy is alive, if not end game with win-flag 995
9   600 increase round\_counter + 1
10  700 reset round\_state and loop to 50
11
12 # round-states without looping anymore:
13 990 gaveover / game lost
14 995 game is won
15 999 show link to exit

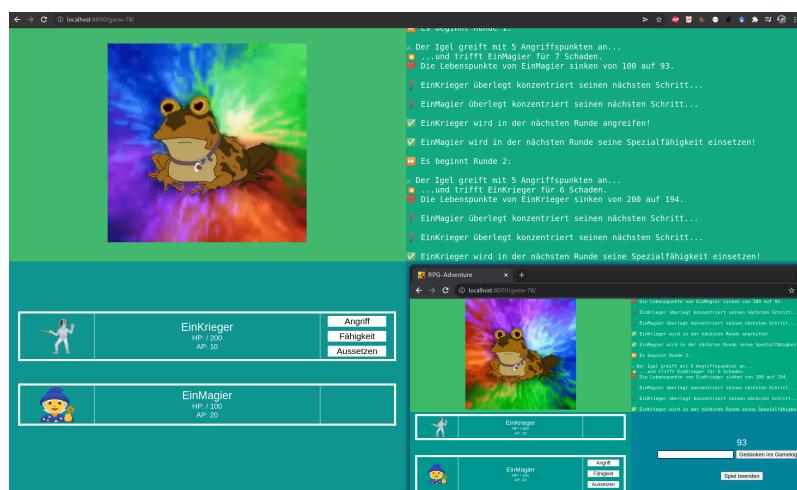
```

Tests ergaben Probleme beim Spiel mit mehreren Spielern. Die Rundenlogik wurde gleichzeitig vorangetrieben. Dadurch wurden manche Aktionen und Rundenschritte mehrfach ausgeführt. Ein Versuch das Problem mit einem Token (ähnlich einem Semaphor) zu lösen, brachte leider keinen abschließenden Erfolg. Der Einzelspieler aber funktionierte fehlerfrei.

Als Workaround für oben genanntes Problem im Mehrspieler wurde eingestellt, dass immer nur der erste Spieler eines Spiels die Rundenlogik vorantreibt. Das ist etwas fehleranfälliger als eine korrekte Token-Lösung, wird für das Projekt hier aber vorerst ausreichend sein (Commit <https://git.io/Jy1su>).

Die Hauptentwicklungsphase endete mit einer Spielseite, die sich noch farbenfroh darstellte:

**Abbildung 9: Bildschirmfoto der Spieleseite zum Entwicklungsstand per 29.12.2021**



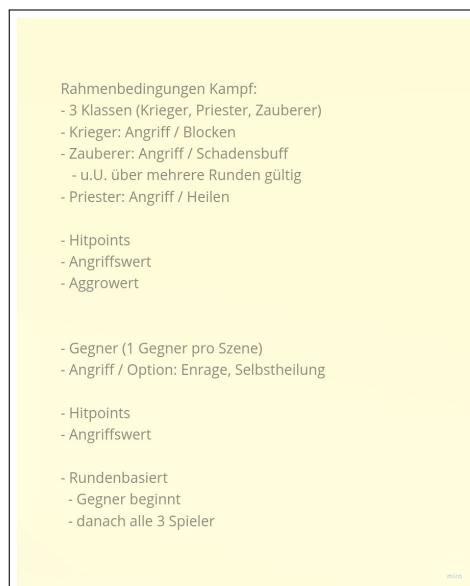
## 5.2 Spieldesign

### 5.2.1 Allgemeines

Am Anfang der Entwicklung musste ein geeignetes Setting für das geplante RPG gefunden werden. Zur Auswahl standen die beiden klassischen, den meisten bekannten Arten von Spieleuniversen. Die eine wäre futuristische Science Fiction und die andere eher mittelalterliche Fantasy.

In der frühen Phase der Entwicklung war beim Spieldesign ursprünglich geplant ein RPG zu entwickeln, dass viele Elemente des klassischen Pen-and-Paper Rollenspiels, wie z.B. Dungeons & Dragons enthält. Man wollte unterschiedliche Charakterklassen, die sich in ihrer Ausrüstung, wie individuellen Waffen und Rüstungen, sowie ihren spezifischen Fähigkeiten, klar voneinander unterscheiden. Ein Levelsystem, bei dem die einzelnen Charaktere von Abenteuer zu Abenteuer ihre Fähigkeiten verbessern und bessere Ausrüstung in Form von Beutegut finden können, sollte auch enthalten sein. Das Entwicklerteam hat sich dann für das Fantasy-Setting entschieden, weil man der Ansicht war, dass sich die vorher genannten Eigenschaften damit besser umsetzen lassen. Speziell die unterschiedlichen Eigenschaften der Charakterklassen waren bei dieser Entscheidung maßgeblich. Denn eine Eingrenzung auf die besonderen Fähigkeiten der unterschiedlichen Spielfiguren, wie z.B. Tank, Heiler und Damagedealer, schien im Science Fiction-Setting nicht so einfach und für den Spieler nicht schlüssig zu gestalten.

**Abbildung 10: Details den Klassen aus der Projektbesprechung vom 11.12.2021**



Die Überlegungen führten somit schon am Anfang recht schnell zu den bereits erwähnten unterschiedlichen Spielfiguren, die der Spieler verkörpern kann. Ein Tank soll Schaden von anderen und sich selbst abhalten können. Ein Heiler soll sich und seine Mitspieler heilen können. Ein Damagedealer soll seinen Schaden und den der anderen Spielfiguren erhöhen. Es war angedacht, dass die einzelnen Spielfiguren über Lebenspunkte, Magiepunkte und verschiedene Handlungsoptionen verfügen, die sich, abhängig von der Charakterklasse, voneinander unterscheiden. Auch schien klar zu sein, dass die Würfelmechanik, die Pen-and-Paper Rollenspielen zu Grunde liegt, in das Spiel übernommen werden soll.

Das Spiel sollte vom Ablauf her, aus aufeinander aufbauenden Abenteuern bestehen, die die Charaktere gemeinsam erleben und die mit Hilfe von unterschiedlichen Ansätzen gelöst werden können, bestehen. Dazu sollten sie sich gemeinsam durch eine grafisch animierte Spielwelt bewegen.

Es sollte Regeln geben, die bestimmen wie und in welcher Reihenfolge der Kampf abläuft oder zu welchen Bedingungen z.B. ein Schwerthieb trifft. Dies sollte auf Grundlage der Open Source Lizenz des Rollenspiels Dungeons & Dragons 3.5 basieren.

Nachdem dieser grobe Rahmen festgelegt wurde, entschieden wir zunächst einmal einen Prototypen zu entwickeln, der die grundlegende technische Machbarkeit unserer Ideen beweisen sollte.

Dabei fiel recht schnell auf, dass vor allem die geplanten Unterschiede der einzelnen Charakterklassen, in Form von sich unterscheidenden Handlungsmöglichkeiten in der Szene, nicht klar abzugrenzen sind. Außerdem war nicht klar, wie es zu handhaben ist, wenn der Charakter z.B. vor einer verschlossenen Tür steht und jede Klasse eine andere Möglichkeit hat dieses Hindernis zu überwinden.

Bei diesem Beispiel ist recht einfach zu klären, wie die unterschiedlichen Handlungsmöglichkeiten aussehen könnten: "Tür eintreten", "Schloss knicken" und "Tür mit Magie öffnen". Dabei war allerdings nicht klar, wie genau dies ablaufen sollte. Wer z.B. darf in diesem Moment die Handlung bestimmen? Derjenige, der zuerst auf den Button klickt? Und was geschieht danach, entsprechend der unterschiedlichen Auswahl der Möglichkeiten? Sollte jede Aktion zu unterschiedlichen Ergebnissen führen oder zu denselben? Dieses würde wiederum die Auswahl, was zu tun ist, völlig unnötig machen. Und falls die Auswahl zu unterschiedlichen Ergebnissen führt, entsteht ein für uns nicht einzuschätzender Aufwand an Storytelling, grafischen Animationen und an Programmierarbeit. Außerdem ist die Komplexität der Kampfmechanik, wie sie in dem Regelsystem von Dungeons & Dragons 3.5 geregelt ist, recht anspruchsvoll und für einen gemütlichen Abend mit seinen Freunden und auf das Spielen von Angesicht zu Angesicht ausgelegt.

Dies ermöglicht eine andere Spielweise als allein vor einem PC. Zum Beispiel kann sich ein Charakter aus dem Kampf zurückziehen oder ein anderer aus der Abenteurergruppe nimmt seinen Platz in der Kampflinie ein. Der Spielleiter, der die Gegner steuert, kann auch entscheiden einen anderen Charakter anzugreifen, wenn er den Eindruck hat, dass der Charakter sonst stirbt. Dies sind Mechaniken, die zwar zu programmieren möglich sind, aber den Rahmen für das erste Projekt dieser Art sicher sprengen würden. Außerdem besteht ein Charakter in diesem Regelwerk aus vielen Eigenschaften, die in Zahlenwerten festgehalten werden. All diese Werde beeinflussen wie er z.B. kämpfen, stehlen oder mit anderen Menschen interagieren kann. Jeder Gegenstand, der als Beutegut von den Charakteren gefunden wird, hat Auswirkungen darauf. Dies ist nur eine grobe Zusammenfassung der Dinge, die die Regelmechanik beeinflussen. Auch große Spieletitel wie z.B. *Baldur's Gate* von BioWare, die ebenfalls auf Dungeons & Dragons 3.5 basieren, haben nicht das gesamte Regelwerk übernommen.

Aufgrund all der oben genannten Unwägbarkeiten und Komplexität, haben wir uns an diesem Punkt zu einer im Folgenden erklärten Konzeptänderung entschieden.

Der gravierendste Schritt ist sicher, dass aus dem RPG-Adventure ein stärker auf Kämpfe ausgerichtetes Rollenspiel mit Fantasy-Setting und atmosphärischen Texten und Grafiken geworden ist. Dabei gibt es zwar noch immer einzelne Level, die man durchspielt, diese sind aber recht linear und bestehen nur aus einem Kampf gegen einen Gegner. Ganz grob soll es wie folgt ablaufen:

Am Anfang der Szene wird ein Introbild mit Ambienttext eingeblendet und nachdem die oder der Spieler bestätigt haben, wechselt die Ansicht. Nun sieht man eine Abbildung der Szene mit dem Gegner. Rechts davon befindet sich das Kampffeld links darunter die Charakteranzeige und ganz unten das Chatfenster. Je nach Ausgang des Kampfes wird ein anderer Outrotext angezeigt und man springt zurück auf die Weltkarte. Es wird keine Ausrüstung geben, die die Charaktere finden können, denn jeder gefundene Gegenstand sollte unterschiedliche Eigenschaften haben und sich in irgendeiner Weise auf den Charakter auswirken. Dies hätte zu diesem Zeitpunkt der Entwicklung zu einem zu umfangreiches Datenbankmanagement geführt. Das Konzept, dass ein Charakter seine Eigenschaften verbessert, in dem er bei den Kämpfen an Erfahrung gewinnt, besteht weiterhin, wenn auch etwas einfacher als zunächst angedacht. Auch ist es so geblieben, dass sich jede Charakterklasse von der anderen durch eine Spezialfähigkeit unterscheidet. Die Klassen sind ebenfalls geblieben, nur hat sich die Bezeichnung etwas konkretisiert. Im Einzelnen sind das der Krieger, der die Möglichkeit hat den erhaltenen Schaden zu reduzieren. Dann gibt es noch den Priester, der die Fähigkeit hat zu Heilen, während der Magier den Schaden erhöhen kann. Nach jedem Kampf gibt eine individuelle Summe

an Erfahrung, die dem Charakter gutgeschrieben wird. Diese kann er dazu benutzen die Fähigkeiten seines Charakters zu verbessern. Der Spieler kann mit der erhaltenen Erfahrung die Lebenspunkte (HP) oder die Angriffskraft (AP) seines Charakters steigern. Ein Charakter erhält immer Erfahrung, egal ob der Kampf gewonnen wird oder verloren geht. Bei einem Sieg ist diese jedoch deutlich höher als bei einer Niederlage. Das hat zur Folge das man einen Level eventuell mehrfach spielen muss, um den nächsten Level erfolgreich abschließen zu können. Der Mechanismus des Würfels ist ebenfalls in der Art implementiert, dass nicht jeder Treffer einen festen Schaden zufügt, sondern es einen Minimum- und einen Maximumsschaden gibt, der zufällig zwischen beiden Extremen errechnet wird. Der endgültige Tod eines Charakters ist nicht vorgesehen.

Die Kampfmechanik wird wie folgt aussehen: Der Kampf ist deutlich vereinfacht und wird rundenbasiert ablaufen, wobei in jeder Runde der Gegner zuerst angreift. Danach wird jeder Charakter die Möglichkeit haben entweder anzugreifen oder seine Spezialfähigkeit einzusetzen. Aussetzen ist ebenfalls möglich. Falls der Charakter seine Spezialfähigkeit nutzt, kann er nicht angreifen und umgekehrt. Die Spezialfähigkeit wirkt mehrere Runden nach. Es ist nicht vorgesehen zu testen, ob ein Angriff trifft oder nicht, ein Angriff trifft also immer und macht Schaden. Der Gegner verfügt über keine Spezialfähigkeiten. Macht dieser Schaden, so wird ein Spieler ausgewählt, dem dann der spezifischen Schaden zugefügt wird. D.h. der Warg der z.B. 20 Punkte Schaden pro Angriff macht, fügt dem ausgewählten Charakter die 20 Punkte Schaden zu. Der Schaden der Spieler summiert sich, sodass drei Spieler, die jeweils 20 Schaden zufügen, dem Warg in Summe 60 Punkte Lebensenergie abziehen. Vor jedem neuen Kampf verfügen die Spieler wieder über ihre vollen Lebenspunkte (HP).

### 5.2.2 Balancing

Trotz der recht einfachen Mechanik ist das Balancing doch recht komplex. Die einzelnen Klassen unterscheiden sich klar in ihren Fähigkeiten voneinander und sollen trotz ihrer unterschiedlichen Spielweise ungefähr gleichwertig sein und dem Spieler natürlich auch gleich viel Spaß bereiten. Um dafür die richtigen Stellschrauben zu haben, sind sämtliche Werte so hinterlegt, dass sie einzeln abgeändert werden können. Im Einzelnen stellt sich das wie folgt dar:

Wie schon erwähnt, können HP und AP für jede Klasse und jeden Gegner einzeln und völlig unabhängig voneinander abgeändert werden. Die Spezialfähigkeiten der einzelnen Charakterklassen sind in Dauer und Ausprägung unterteilt, die sich wie bei den anderen Werten individuell für jede Klasse einzeln regeln lassen. Auch die erhaltene Erfahrung

---

ist mit einem abänderbaren Multiplikator versehen, um Einfluss darauf zu nehmen, wie viel Erfahrung die einzelnen Charaktere erhalten und wie schnell sie ihre Fähigkeiten verbessern können. Dabei ist ebenfalls darauf geachtet worden, dass es die Möglichkeit gibt, dass HP und AP unterschiedlich viele Erfahrungspunkte kosten können und wie alle anderen Werte ist dies auch variabel einstellbar. Die unterschiedlichen Erfahrungspunktkosten von HP und AP sind nötig, weil der Unterschied von diesen die Charaktere voneinander abgrenzt und diese im Kampf unterschiedlich wichtig sind und die Spieler nicht zu schnell zu mächtig werden sollen.

```

1 "name": "ability_m_effect_strength",
2 "type": "float", "value": "0.6",
3 "hint": "strength of the mage ability in percent (written in float, 0.1 = 10%,
        0.5 = 50%)"
4
5 "name": "ability_m_duration_rounds",
6 "type": "int", "value": "2",
7 "hint": "mage ability will be applied to this number of next rounds"
8
9 "name": "ability_p_effect_strength",
10 "type": "float", "value": "0.08",
11 "hint": "strength of the priest ability in percent (written in float, 0.1 = 10%,
        0.5 = 50%)"
12
13 "name": "ability_p_duration_rounds",
14 "type": "int", "value": "4",
15 "hint": "priest ability will be applied to this number of next rounds"
16
17 "name": "ability_w_effect_strength",
18 "type": "float", "value": "0.20",
19 "hint": "strength of the warrior ability in percent (written in float, 0.1 = 10%,
        0.5 = 50%)"
20
21 "name": "ability_w_duration_rounds",
22 "type": "int", "value": "3",
23 "hint": "warrior ability will be applied to this number of next rounds"
```

Die Trennung von HP und AP hat außerdem zur Folge, dass nicht jeder Charakter derselben Klasse demselben Stereotyp entspricht. Also ist es möglich, dass sich jeder Krieger, so der Spieler denn möchte, anders entwickeln kann als der Krieger, den der Spieler davor gespielt hat. Der Spieler kann also einen Krieger erschaffen, der entweder Wert auf HP oder AP legt und das jedes Mal individuell neu entscheiden. Dies gilt auch für alle anderen Klassen. Der variable Schaden soll dazu dienen, dass nicht jeder Kampf vorherberechnet werden kann, indem man die HP des Gegners mit den AP des Spielers vergleicht.

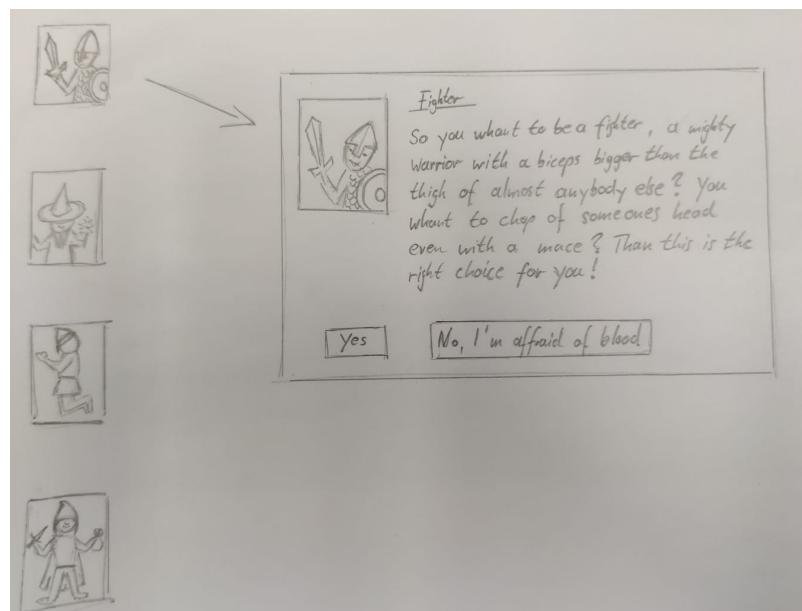
All das führt dazu, dass das Balancing gut und kleinschrittig angepasst werden kann, macht es aber aufgrund der vielen Stellschrauben auch sehr komplex und für uns, die wir über wenig Erfahrung im Spielebalancing verfügen, auch schwierig alles aufeinander

abzustimmen. Die im veröffentlichten Spiel festgelegten Werte stellen einen Kompromiss aus Arbeitsaufwand und Spielbarkeit bzw. Spielerfahrung dar. Diese könnten noch durch ein umfangreiches Beta-Testing, das üblicherweise an so einem Punkt bei Spieleentwicklungen geschieht, optimiert werden. Dafür gibt es hier aber weder Zeit noch Ressourcen, um die Rückmeldungen von einer Vielzahl von Testern auszuwerten und in die Entwicklung einfließen zu lassen und anschließend nochmals zu Prüfen ob die Änderungen zu den gewünschten Ergebnissen geführt haben. Beim Balancing des Spiels hat sich außerdem gezeigt, dass die Teilbereiche Programmierung und Balancing unbedingt eng zusammenarbeiten sollten. So kann sichergestellt werden, dass die nötigen Mechaniken auch vorhanden sind und nicht noch nachträglich evtl. umständlich implementiert werden müssen. Wenn dies dann überhaupt noch möglich ist.

### 5.2.3 Storytelling

Am Anfang war generell festzulegen in welcher Sprache das Spiel erscheinen soll, dabei standen Englisch und Deutsch zur Auswahl. Ursprünglich sollte das Spiel auf Englisch erscheinen und die ersten Konzepttexte der Charakterbeschreibungen wurden auch in englischer Sprache verfasst (zu sehen in folgender Abbildung).

**Abbildung 11: Entwurfszeichnung und Text für eine Charaktererstellungsseite (29.11.2021)**



Da allerdings jeder der Projektteilnehmer deutscher Muttersprachler ist, wurde die Projektsprache Deutsch festgelegt. Außerdem wird das Spiel nur im Zusammenhang eines

Studienprojektes entwickelt und soll im deutschsprachigen Raum veröffentlicht werden. Um die Entwicklung nicht unnötig auf Grund von sprachlichen Schwierigkeiten zu verkomplizieren, schien dieser Schritt logisch.

Die Entwicklung hin zu einem Spiel, in dem die Level oder Spielszenen nur lose zusammenhängen, machte es nötig zu jeder einzelnen Szene eine Story zu schreiben, um dem Spieler ein Gefühl zu geben, was gerade passiert und warum. Außerdem sind unterschiedliche Texte je nach Ausgang der Spieleszene vorgesehen und Texte, die z.B. beschreiben was gerade im Kampf geschieht. Dabei ist es wichtig sich auf die Beschreibung der dargestellten Szene zu konzentrieren und nicht abzuschweifen, da ein zu langer Text vom Spieler vielleicht nicht gelesen oder als störend empfunden wird. Jedoch muss er lang und intensiv genug sein, damit sich der Spieler einen Eindruck von dem Geschehen verschaffen und darin eintauchen kann. Schließlich soll das Spiel auch eine Geschichte erzählen und dem Spieler eine gewisse Spielerfahrung und im Idealfall einen Wiederspielwert geben.

Beim Entwickeln der einzelnen Szenen ist aufgefallen, dass man nicht immer genau sagen kann, was zuerst da war, das Huhn oder das Ei, denn der Text und die Grafik stehen in engem Zusammenhang und haben sich gegenseitig beeinflusst. Die Beschreibung der Szene stand üblicherweise zuerst und danach wurde die Szenengrafik entwickelt. Aber manchmal war es auch umgekehrt oder es war nötig den Text anzupassen, weil die Animation z.B. eines Rudels Wölfe schwieriger war als die von einem einzigen riesigen Wolf. So wurde aus dem Rudel Wölfe, das die Gegend terrorisiert, ein einzelner großer Warg. Oder im Fall der Szene im Anwesen war bei ursprünglicher Planung ein Geist als Gegner vorgesehen, aber beim Schreiben der Geschichte wurde daraus ein Vampir, der viel düsterer ist und besser passt.

Daraus ist abzuleiten, dass bei einer Spieleentwicklung diese beiden Teilbereiche sehr eng zusammenarbeiten sollten. Für den Programmierer z.B. ist es egal, was für eine Grafik oder Text an entsprechender Stelle im Code eingefügt werden soll. Oder um es in den Worten unseres Lead-Programmers auszudrücken: „Ich bin da total leidenschaftslos“. Aber Spiel- und Grafik-Design müssen unbedingt Hand in Hand gehen und sich gegenseitig ergänzen, um eine überzeugende Spielerfahrung zu schaffen.

Im Fall des Kampfes gegen den Drachen und den Vampirlord ist man bewusst von der reinen Beschreibung der Szene abgewichen und hat viel mehr die Motivation des Charakters und etwas Hintergrundgeschichte in den Vordergrund gestellt, um beim Spieler die Beweggründe des Charakters in den Vordergrund zu stellen. So kann sich dieser, nicht wie bei den anderen Szenen in die Handlung, sondern mehr in den Akteur selbst hineinversetzen. Bei diesen Gegnern sollte auch etwas Besonderes im Text stehen, außerdem wollte der Autor auch unterschiedliche Arten der Erzählung ausprobieren.

### 5.3 Grafiken und Animationen

Jede der Spielszenen hat drei Grafik-Assets: eine Intrografik für die Szenenlobby, einen Hintergrund für den Kampfbildschirm und eine Bossgegner-Grafik mit 2D-Animationen. Zudem kommt noch eine weitere Grafik für die Weltkarte hinzu. Diese Letztere, Intro und Hintergrund besitzen das JPG-Bildformat. Die Bossgegner-Grafik dagegen ist ein MP4-Video. Für das JPG-Format entschieden wir uns hauptsächlich aufgrund der im Vergleich zu PNG geringeren Dateigröße. Da es sich bei allen Grafiken um rein statische Bilder handelt, erschien der Komprimierungsverlust von JPG vernachlässigbar. Zudem wurde für die Bilder keine Hintergrund-Transparenz benötigt. Alle Grafiken wurden in einer Auflösung von 3840x2160 (Ultra-HD) gezeichnet und dann entweder so verwendet oder auf 1920x1080 (Full-HD) herabskaliert.

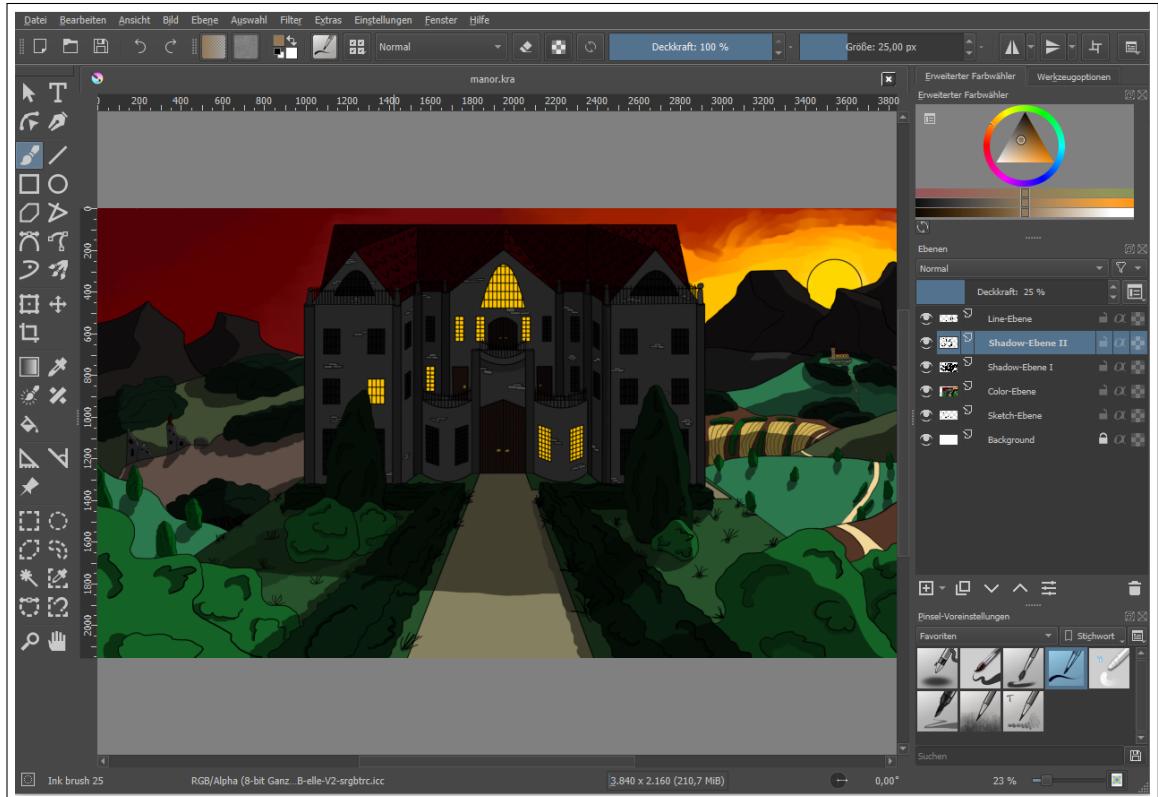
**Abbildung 12: Vom Concept Art zur finalen Grafik**



Fortsetzung siehe Folgeseite.

Für die Erstellung der Grafik-Assets und Animationen wurden das freie Zeichen- und 2D-Animationsprogramm *Krita*<sup>13</sup> und ein Grafiktablet verwendet (in der folgenden Abbildung beispielhaft das Standard-UI von Krita).

**Abbildung 13: Krita Standard-UI**



Die Grafiken wurden hierbei mithilfe der softwareeigenen Ebenenhierarchie systematisch wie folgt aufgebaut:

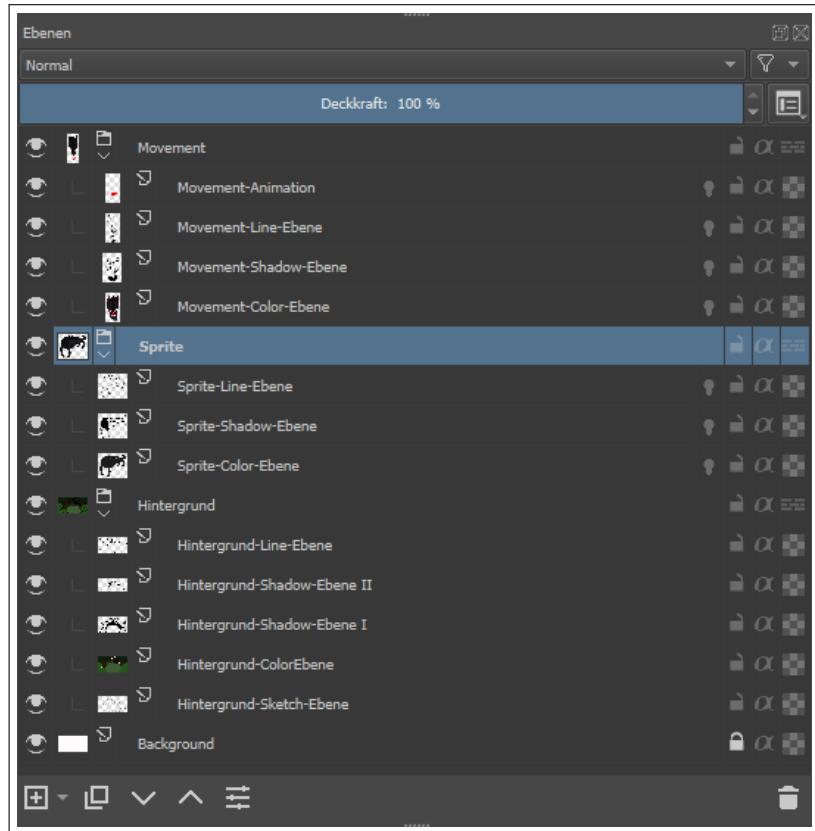
Eine Konzeptzeichnung bildet die unterste Ebene mit einer vergleichsweise geringen Deckkraft („Sketch-Ebene“). Darüber folgt die „Linien-Ebene“, auf der auf Basis der Skizze eine vollständige Strichzeichnung mit maximaler Deckkraft angefertigt wird. Diese bildet die oberste Ebene und alle folgenden Ebenen werden aufsteigend zwischen diesen beiden positioniert. Durch eine sorgfältige Linienführung auf dieser Ebene kann das Fülltool auf der nun folgenden „Farb-Ebene“ effektiv genutzt werden. Den Abschluss bilden ein oder zwei „Schatten-Ebenen“, die mit einer Deckkraft von 50% bzw. 25% über der eigentlichen Kolorierung liegen.

Im Falle der Bossgegner-Grafiken wird dieses Konzept noch um drei Gruppen ergänzt, welche jeweils alle Ebenen in sich zusammenfassen. Es gibt eine Gruppe für das Hintergrundbild, eine Gruppe für die nicht-animierten Teile des Bossgegners und eine Gruppe

<sup>13</sup> Siehe <https://krita.org/en/>.

für die eigentlichen Animationen. Zum Vergleich ist im Folgenden eine Abbildung dieses Aufbaus zu sehen (hier beispielhaft eine Grafik mit 2D-Animation):

**Abbildung 14: Ebenenkonzept für die Grafiken**

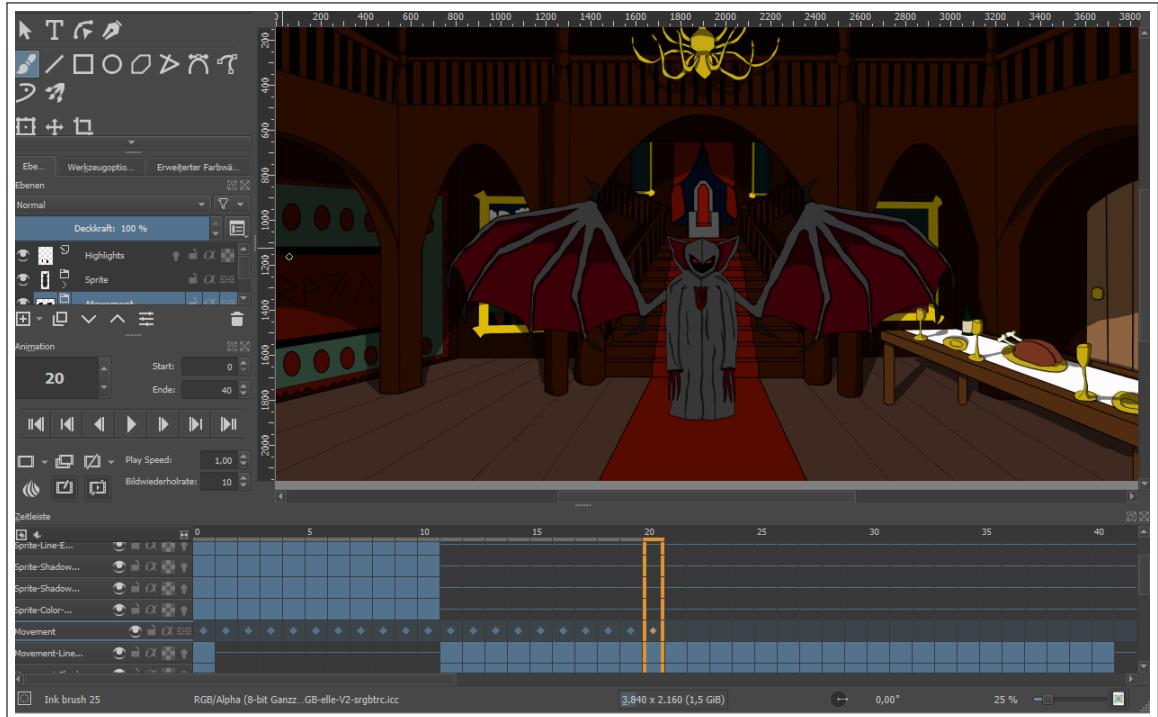


Dieses Konzept brachte eine erhebliche Flexibilität mit sich. Es war uns damit möglich effizient Änderungen an separaten Teilen der Grafiken durchzuführen, mit der Farbgebung zu experimentieren und nachträglich neue Details einzubringen. Zudem konnten Hintergrundbilder und Bossgrafiken als eine Datei erstellt werden und dann durch Ausblenden verschiedenener Ebenen individuell exportiert werden. Auch der Animationsprozess wurde durch diese Technik deutlich erleichtert.

Für die Realisierung der 2D-Animationen bietet *Krita* ein dediziertes User-Interface (siehe Abbildung unten). Die Ebenen können hier je nach Bedarf zusammen oder einzeln animiert werden. Dazu werden sie unabhängig voneinander in die Animationsframes kopiert und gegebenenfalls verändert. Im unten zu sehenden Beispiel werden die Ebenengruppen „Sprite“ (nicht-animierter Teil der Bossgegner-Grafik) und „Movement“ (eigentliche Animation) über 10 Frames hinweg durch sukzessive Erhöhung der Deckkraft eingebettet, um den Effekt eines plötzliches Erscheinen aus Unsichtbarkeit zu erzielen. Danach wird in der „Movement“-Gruppe über 29 Frames eine Flügelbewegung realisiert. Die

Animation wurde dann mit einer Bildwiederholrate von 10 Frames per Second gerendert und als MP4-Video in Ultra-HD und Full-HD exportiert.

**Abbildung 15: Krita Animation UI**



Die Separierung der Grafiken in animierte und nicht-animierte Teile, stellte bei der eigentlichen Animation eine erhebliche Arbeitserleichterung dar. Viele Bewegungsabläufe konnten durch Verschieben vorher definierter Abschnitte erreicht werden und mussten nicht pro Frame neu gezeichnet werden (teilweise wurde dies aber trotzdem nötig). Trotz dieser Erleichterung ergab sich aber immer noch ein erheblicher Arbeitsaufwand selbst für nur wenige Frames. Daher entschieden wir es bei vergleichsweise rudimentären Animationen zu belassen. Eine Realisierung in 30 Frames per Second beispielsweise wäre für das Projekt unverhältnismäßig zeitaufwendig gewesen.

Für die Ausgestaltung des Game-Logs und verschiedener Ambienttexte und die Darstellung der Klassen wurden vorgefertigte Emojii-Symbole von der Website [emojiterra/footnoteSiehe https://emojiterra.com](https://emojiterra.com). verwendet.

## 5.4 Frontend-Design

Das im Rahmen dieses Projekts entstandene Spiel ist von klassischen Text-Adventures und Rollenspielen der Achtziger- und Neunzigerjahre inspiriert. Um den speziellen Look dieser Spiele zu treffen, entschieden wir uns, handgezeichnete Grafiken und eigene 2D-Animationen anzuwenden. Das Frontend-Design des Browserspiels sollte ebenfalls eine Hommage an diese Ära sein und dabei helfen ihren besonderen Charme einzufangen.

Daher wählten wir das CSS-Framework *RPGUI* von Ronen Ness<sup>14</sup> als Basis der Gestaltung der Browserspiel-Oberfläche aus. Dieses Framework liefert eine große Anzahl vorgefertigter Klassen für verschiedenste HTML-Elemente im „Old-School-Look“ klassischer 8-Bit und 16-Bit-RPG's. Neben dem eigentlichen CSS-Stylesheet werden zudem viele JavaScript-Funktionen bereitgestellt, die z.B. die Benutzung von dynamischen Elementen wie einer Lebenspunkte-Anzeige ermöglichen.

**Abbildung 16: Frontend-Design Beispiel: Szenenlobby**



Für die Implementation in das Frontend wurde zunächst dem übergeordnetem div-Container in der *base.html* die Klasse *rpgui-content* zugeordnet. Dort sind grundlegende CSS-Stylings enthalten, die z.B. Schriftart, Schattierung, Hintergrundfarbe und Ausrichtung der Hintergrundbilder für die einzelnen Elemente definieren. Diese werden an weitere Elemente mit der Klasse *rpgui-content-\** vererbt. Mithilfe der Template-Vererbung von Django via **extends** konnte so ein Großteil der Frontend-Gestaltung zentral in der *base.html* vorgenommen werden.

Trotzdem war es in vielen Fällen notwendig Anpassungen an HTML-Elementen direkt in den Template-Dateien durchzuführen. Insbesondere für die Platzierung der Elemente

<sup>14</sup> Siehe <https://github.com/RonenNess/RPGUI>, veröffentlicht 2016 unter der zlib-Lizenz

auf der Webseite und die korrekte Skalierung für verschiedene Bildschirmgrößen wurde stark auf Inline-CSS zurückgegriffen. Dies sieht im Code dann wie folgt aus (Beispiel aus *game\_endscreen.html*):

```
1 <div class="rpgui-container framed-golden"
2     style=" width: 50%;
3             font-size: 1.4vw;
4             align-items: center;
5             text-align: center;
6             justify-content: center;
7             margin-left: 25%;
8             margin-right:25%">
```

Hier wurde die von *rpg-content* festgelegte font-size mit einem Viewport-Wert überschrieben, um eine bessere Skalierung zu ermöglichen. Zudem wurden im style-Element verschiedene Einstellung für das Seitenlayout vorgenommen. Dies wurde in ähnlicher Weise an vielen Stellen im Code so gemacht.

Wie oben erwähnt, wurden die meisten individuellen Einstellungen dafür verwendet, die Elemente skalierbar zu machen. Eines unserer Ziele für das Projekt war es, das Frontend zumindest bis zu einem gewissen Grad so zu gestalten, dass eine geräteübergreifende Nutzbarkeit des Spiels möglich ist. So sollte beispielsweise innerhalb des Kampfbildschirms kein Scrollen notwendig sein. Es sollte sichergestellt werden, dass wichtige Steuerelemente, wie z.B. die Aktions-Buttons und der Game-Log, zu jedem Zeitpunkt vollständig sichtbar sind. Ein Scrollen und Suchen nach diesen Elementen erschien uns in Anbetracht der „Action-Szene“, in der sich die Spieler an diesem Punkt befinden, als der Immersion sehr abträglich. Daher wurde das overflow-Attribut für die gesamte *game.html* und deren Tochter-Templates *game\_content.html* und *game\_endscreen.html* deaktiviert und das Layout entsprechend gestaltet.

Hierfür war es zum Teil auch notwendig bestehende CSS-Klassen aus dem Framework anzupassen. Ein Nachteil des RPGUI-Frameworks ist die häufige Angabe von fixen Pixel-Maßen, die ein skalierbares Layout behindern. Im folgenden Code-Beispiel wird exemplarisch die angepasste RPGUI-Klasse *rpgui-container.framed-scalabe* gezeigt, welche aus der Originalklasse *rpgui-container.framed* erstellt wurde:

```
1  /* Skalierbare Elemente für game.html */
2   .rpgui-container.framed-scalable {
3  /* border */
4    border-style: solid;
5    border-image-source: url("img/border-image.png");
6    border-image-repeat: repeat;
7    border-image-slice: 6 6 6 6;
8    border-image-width: 0.5vw;
9    border-width: 0.4vw;
10   padding-top: 0.8vh;
11   padding-bottom: 0.8vh;
12   padding-left: 0.8vh;
13   padding-right: 0.8vh;
14  /* internal border */
15   box-sizing: border-box;
16   -moz-box-sizing: border-box;
17   -webkit-box-sizing: border-box;
18  /* background */
19   background: url("img/background-image.png") repeat repeat;
20   background-clip: padding-box;
21   background-origin: padding-box;
22   background-position: center; }
```

Das Ziel der Skalierbarkeit haben wir grundsätzlich erreicht, es gibt allerdings gewisse Grenzen. Das UI wird auf Bildschirmen mit Tabletmaßen noch annehmbar skaliert - die für das Spielen nötigen Elemente sind bedienbar. Für das Spielen über ein Smartphone ist das Spiel allerdings nicht geeignet, da hier Elemente und Layout zu stark verschoben werden. Einer der Gründe hierfür ist die Verwendung von Viewport-Maßen (vh = viewport-height, vw = viewport-width; vgl. obiges Code-Beispiel) in unserem Frontend-Code. Diese Maße orientieren sich in ihrer Skalierung am Viewport des jeweiligen Gerätes (Viewport = Sichtfenster/Sichtöffnung/Display) und sorgen für eine theoretisch sehr effektive geräteübergreifende Anpassung. Die Angabe von fest vorgegebenen Viewport-Maßen im style-Element führt allerdings zu einer teilweise zu starken Verkleinerung bzw. Vergrößerung. Es müssten hier eigentlich individuelle Grenzen für jede Gerätgruppe gesetzt werden, was über sogenannte „Media Queries“ erreicht werden könnte. Uns ist diese Technologie bekannt, aber wir entschieden uns es trotzdem aus Aufwandsgründen bei der eher rudimentären geräteübergreifenden Skalierung zu belassen. Unser Hauptfokus lag während der Entwicklung eindeutig auf Notebook- und Desktop-Rechner Bildschirmgrößen und auf diesen verhält sich das UI korrekt.

In seltenen Fällen wurden von uns auch vollständig neue CSS-Klassen erstellt. Für den Spieldaten links oben in der Navigationsleiste wurde z.B. eine eigene Klasse *game\_title* geschrieben. Durch die feste Vorgabe von grundlegenden Attributen wie font-size und color durch die *rpgui-content* konnten wir den Titel nicht individuell formatieren. Auch Inline-CSS wird für diese Attribute überschrieben. Die Lösung war eine eigene Klasse zu erstellen und deren Attributvorgaben mit **!important** zu kennzeichnen, was dazu führt,

dass sie auch bei konkurrierenden Angaben eigentlich übergeordneter Klassen durchgesetzt werden.

```
1 .game_title{  
2     font-size: xx-large !important;  
3     color: #8B0000 !important;  
4     text-decoration: underline !important;  
5     /*border-style: solid !important;  
6     border-color: #8B0000 !important;  
7     border-radius: 25px !important;  
8     border-width: 7px !important;*/  
9     padding-top: 4px !important;  
10    padding-bottom: 4px !important;  
11    padding-left: 11px !important;  
12    padding-right: 4px !important;  
13    white-space: nowrap; }
```

## 6 Reflektion

An dieser Stelle finden sich stichpunktartig einige Reflektionen von uns, die sich zum Abschluss der Arbeit ergaben und aus unserer Sicht hier durchaus nennenswert sind.

**Rundenbasierter vs. chaotischer Spielablauf:** Der Ansatz die Entwicklung durch den Einsatz eines rundenbasierten Ablaufs bzw. Kampfsystems deutlich zu vereinfachen, musste im Laufe der Entwicklung zumindest angezweifelt werden. Denn der Aufwand, der durch die notwendige Konzeptionierung und Detailplanung entsteht, ist nicht zu unterschätzen. Dementgegen stünde bei einem chaotischen Spielablauf lediglich das Handling der Events.

**Kleinteilige Aufgabenpakete:** In der Entwicklung eine überschaubare Anzahl an kleinteiligen bzw. Teilaufgaben vor sich zu haben, empfanden wir als sehr hilfreich. Man hat damit einen Überblick über die Arbeit der nächsten Tage. Bei der Entwicklung der Rundenlogik entstand durch die Kenntniss um die nächsten Rundenschritte hier ein stets guter Überblick. Der Abschluss jeder einzelnen Teilaufgabe sorgte weiter laufend für positive Motivation.

**Lernkurve und Codequalität:** Eigentlich müsste am Ende eines Entwicklungsprojektes stets noch einmal von vorne anfangen. Allein um alles zu korrigieren und anzupassen bzw. auf einen gleichen Quaitätsstand zu bringen, was im Projektverlauf bei den Beteiligten an Fähigkeiten und Wissen gelernt wurde.

**Setting:** Rückwirkend betrachtet, scheint die Wahl des Settings, abhängig von Spielmechanismen, die nachher nicht umgesetzt wurden, nicht mehr so entscheidend, wie zu dem Zeitpunkt der Entwicklung. Denn so wie das Spiel jetzt aufgebaut ist, hätte es durchaus auch als Science Fiction funktionieren können. Es müssten lediglich der optische Auftritt und die Geschichten abgeändert werden. Der technische Unterbau, speziell der des Kampfmechanismus, welcher mitentscheidend für die Änderung war, ist zum gegenwärtigen Zeitpunkt sehr universell und einfach gehalten und könnte so ohne Änderungen übernommen werden.

**Storytelling/Szenenentwicklung:** Damit ist das Team im Großen und Ganzen sehr zufrieden. Die Zusammenarbeit zwischen Autor und Artist war engmaschig und hat gut funktioniert. Es war jedoch nicht immer einfach, nicht zu weit abzuschweifen oder die Szene zu komplex werden zu lassen, denn der Autor hätte sich manchmal noch mehr Tiefe gewünscht, um all seine Ideen umzusetzen. Außerdem merkt der Spieler im fertigen Spiel nicht, welcher Aufwand betrieben wurde, um die Texte und die Szenen zu entwickeln. Für das anvisierte Ziel, welches zu Anfang des Projektes gesteckt wurde, nämlich technische

Umsetzbarkeit und Implementierung aller Anforderungen, war der Umfang vielleicht jetzt schon etwas viel. Denn der Aufwand, im Besonderen im Grafikdesign, war schon recht hoch.

**Balancing:** Das Balancing funktioniert grundsätzlich gut. Die Tatsache, dass der Charakter seine Erfahrung getrennt für HP und AP ausgeben und dadurch eine extreme Schere zwischen den Charakteren entstehen kann, macht das Balancing aber auch recht schwer. Hier wäre es vielleicht mit einem Ansatz, mit festen Erfahrungspunkten je Gegner und ein starres Levelsystem, bei dem man bei festgelegten Grenzen einen Level aufsteigt und feste HP und AP Zuwächse je Level bekommt, einfacher gewesen. Dadurch ließen sich die Gegner in den aufeinander folgenden Leveln viel besser skalieren, gleichzeitig würde aber die Variabilität und Individualität der Charaktere stark beschnitten werden.

**Grafiken & Animationen:** Der Arbeitsaufwand für die Erstellung der Intro- und Hintergrundgrafiken und die Animation der Gegner, wurde von uns zu Beginn der Entwicklung stark unterschätzt. Selbst ohne die Zeit für die Bewältigung von einhergehenden Problemen, wie z.B. dem Erlernen halbwegs vorzeigbarer Zeichentechniken, mitzuzählen, waren viele Stunden notwendig, um alle Assets fertigzustellen. Auf der anderen Seite machen aber gerade Grafikdesign und -stil einen sehr großen Teil des Charmes und der Identität eines Spiels aus. Es ist nun aber besser nachzuvollziehen, warum sich bei der Entwicklung von Spielen durch professionelle Studios ganze Abteilungen um nichts anderes kümmern als um Concept Art, Animation und Grafik-Design.

**Verwendung eines CSS-Frameworks:** Zu Anfang der Entwicklung hatten wir angedacht benötigte Assets für die Frontend-Entwicklung vollständig selbst zu erstellen. So war auch die grafische Aufbereitung von HTML-Elementen zunächst etwas, was wir selbst machen wollten. Hier wurde jedoch schnell klar, dass dies einen enormen Aufwand notwendig machen würde und den Rahmen des verhältnismäßig kleinen Projektes sprengen würde. Das hervorragende CSS-Framework RPGUI konnte an dieser Stelle sehr gewinnbringend eingesetzt werden und hat uns beim Frontend-Design sehr geholfen.

**Genereller Lernkurve:** Vor diesem Projekt verfügten die beteiligten Studenten über keine oder nur sehr geringe Erfahrung im Bereich der Webentwicklung. Weder die Abläufe einer Spielentwicklung noch das Django-Framework waren uns näher bekannt. Da am Ende ein einfaches, aber funktionierendes Spiel dabei herausgekommen ist, kann man behaupten, dass die Lernkurve aller Projektmitarbeiter entsprechend steil gewesen ist. Wir werden sicherlich einiges an Erfahrungen und Best Practices aus diesem Projekt für die Zukunft mitnehmen können.

## Anhang

### Anhang 1: Projektnotizen

Austausch und Zusammenarbeit erfolgte auf verschiedenen Plattformen:

- Gezeichnet und Entwürfe wurden meist in Miro<sup>15</sup> erstellt.
- Besprechungen erfolgten meist in Teams<sup>16</sup>.
- Der gemeinsame Code und die Dokumentation wurden auf Github erstellt: <https://github.com/tstsrv-de/rpg>.

### Anhang 2: Ideen für Erweiterungen über diese Projektarbeit hinaus (Ausblick)

Da der Umfang dieser Projekt beschränkt ist, können nicht alle erdachten Funktionen umgesetzt werden. Einige Ideen, sollen aber nicht ungenannt bleiben:

1. Würfel bei Schaden bzw. Angriff implementieren.
2. Aggratabelle und verbundene Funktionen implementieren.
3. Inaktive und getrennte Spieler bzw. Nutzer aus den Chatfunktionen der Weltkarte und Lobby entfernen. Ggf. auch einen Timer für das Spiel anlegen der anderen Spielern anzeigt, wenn ein Spieler inaktiv bzw. getrennt vom Server ist.
4. E-Mail Funktion von Django konfigurieren, damit das Zurücksetzen von Passwörtern u.A. möglich wird.
5. Der Traefik-Proxy sollte noch um die Monitoring- und Logging-Funktionen erweitert werden.

---

<sup>15</sup> Siehe <https://miro.com/app/board/uXjVOdN2haQ=/>

<sup>16</sup> Siehe [https://teams.microsoft.com/l/team/19%3aDoBvOwOIC6WNhsL9kO1YFKNtVftU1yBtcEn\\_gcyQtcg1%40thread.tacv2/conversations?groupId=850a22ff-34a2-4fe2-a506-f55ac4d595f8&tenantId=b9b6f99a-a243-422d-ab36-f726574c981a](https://teams.microsoft.com/l/team/19%3aDoBvOwOIC6WNhsL9kO1YFKNtVftU1yBtcEn_gcyQtcg1%40thread.tacv2/conversations?groupId=850a22ff-34a2-4fe2-a506-f55ac4d595f8&tenantId=b9b6f99a-a243-422d-ab36-f726574c981a)

## Anhang 3: Notizen Erstellung Django-Anwendung

Die Django-Anwendung für dieses Projekt wurde anhand der Anleitungen und Versuche aus den Vorlesungen neu erstellt. Dabei wurden die Schritte kurz und in Stichworten notiert. Relevant dazu sind die Commits von Samstag 13.11.2021, siehe <https://git.io/JSq7n> und <https://git.io/JSYH4>. Dies als Anhang hier:

```

1 2. rename .env.example to .env and change settings
2 3. init django with: docker-compose run rpg django-admin startproject rpg .
3 4. edit rpg/settings.py:
4   add:
5     start:
6       import os
7       import environ
8       env = environ.Env()
9       environ.Env.read_env()
10      after 'BASE_DIR':
11        TEMPLATES_DIR = os.path.join(BASE_DIR, 'templates')
12        STATIC_DIR = os.path.join(BASE_DIR, 'static')
13      after STATIC_URL:
14        STATICFILES_DIRS = [
15          STATIC_DIR,
16        ]
17    change:
18      SECRET_KEY to:
19        SECRET_KEY = env('ENV_SECRET_KEY')
20      ALLOWED_HOSTS to:
21        ALLOWED_HOSTS = [env('ENV_ALLOWED_HOSTS')]
22      in Templates, Dirs to:
23        'DIRS': [TEMPLATES_DIR],
24      DATABASES to:
25        DATABASES = {
26          'default': {
27            'ENGINE': 'django.db.backends.postgresql',
28            'NAME': env('ENV_POSTGRES_DB'),
29            'USER': env('ENV_POSTGRES_USER'),
30            'PASSWORD': env('ENV_POSTGRES_PASSWORD'),
31          }
32        }
33
34 5. copy and rename .env.example also to rpg/.env (remeber to copy again @changes)
35 6. update django to new db: 'docker-compose run python manage.py makemigrations'
36   and 'docker-compose run python manage.py migrate'
37 7. create superuser: docker-compose run rpg python manage.py createsuperuser
38 8. create django app: docker-compose run rpg python manage.py startapp rjh_rpg

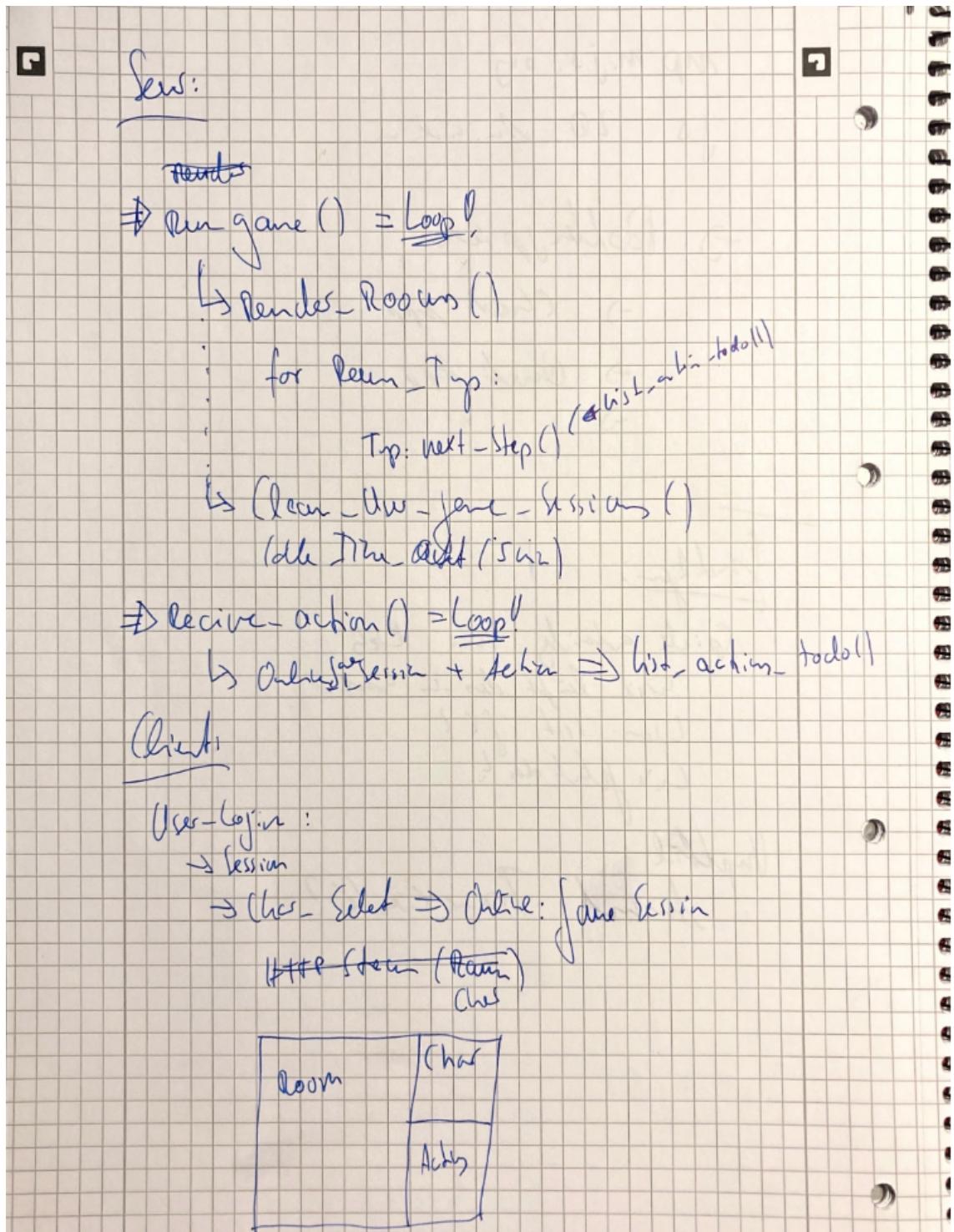
```

## Anhang 4: Inhalte aus und zu Projektbesprechungen

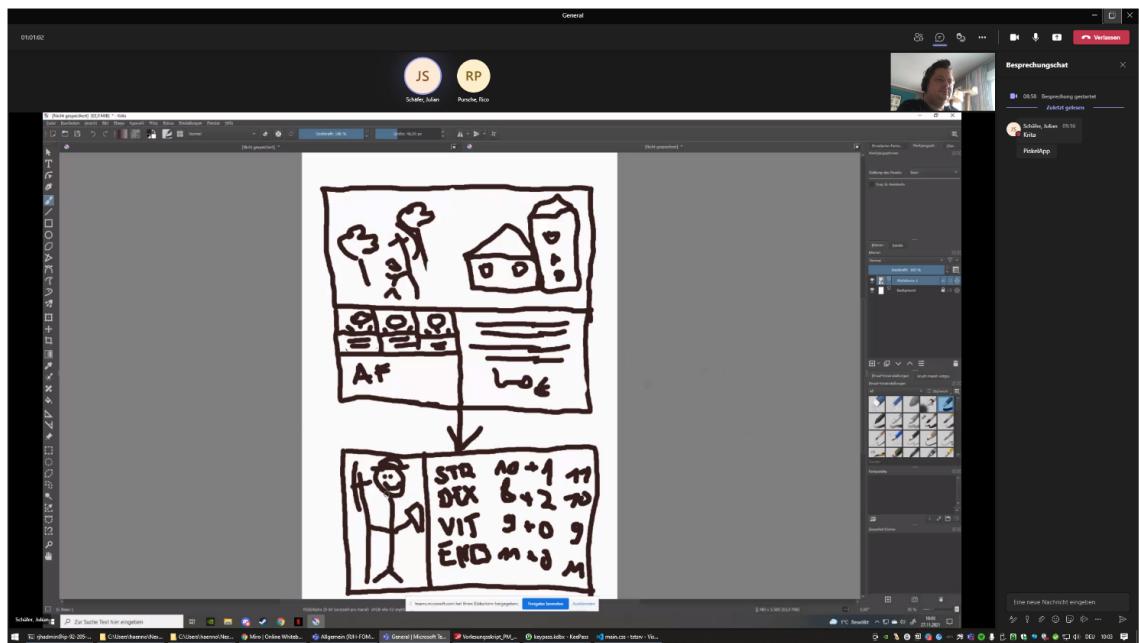
Stets Sonntags erfolgten Projektbesprechungen. Notizen und Zusammenfassungen davon finden sich hier in fortschreitender, chronologischer Reihenfolge (insofern diese im

Hauptteil der Dokumentation nicht bereits dargetellt wurden). Ebenso hier entsprechend eingesortiert, finden sich Konzeptzeichnungen und Entwürfe aller Art (UI, Code, Datenbankmodelle).

**Abbildung 17: Anhang: Allererstes Konzept mit Pseudocode für die Spiel-Logik (23.11.2021)**



**Abbildung 18: Anhang: Erstes Mockup zu einer möglichen UI des Spiels (27.11.2021)**



**Abbildung 19: Anhang: Detailierterer Pseudocode für die Level-Lobby (30.11.2021)**

Lobby - Logik: Abg 30.11. 2021

Lobby - Seite:

- Eigenschaften
  - "Spieleranz." laden
- Bei "last-heatbeat" über Websocket von Spieler-ches:
  - Mögl-Lobby: für alle in Szene: Wieder?
  - (1-1) Wenn "last-heatbeat" nicht 10 Sch
    - Löschen
    - ~~②~~ (2) spieler\_mz (2) pos (Amy bauen + Anna)
      - prüfen locked\_in gecheckt?
    - (2-1) Wenn alle locked\_in klein
    - (2-2) alle auf locked\_in setzen

DS-Modell

```

graph TD
    Lobby[Lobby]
    Lobby --> Spieler[Spiele ich]
    Spieler --> Clear[Clear ich]
    Clear --> Post[Post]
    Post --> DateTime[date time]
    DateTime --> Heatbeat[Heatbeat]
    Heatbeat --> Locked[Locked in]
  
```

(2-1) locked\_in alle als 5 Sek?

↳ Lück zu  
⇒ Spiel au-  
Beginn + feinstufig  
ändern

⇒ Post-heatbeat + Lobby circeln

⇒ locked\_in

⇒ abw EinsLobby in Scene-Modell?

⇒ nur Jane State-Modell

**Abbildung 20: Anhang: Entwurf einer UI für die Lebel-Lobby mit Details (30.11.2021)**

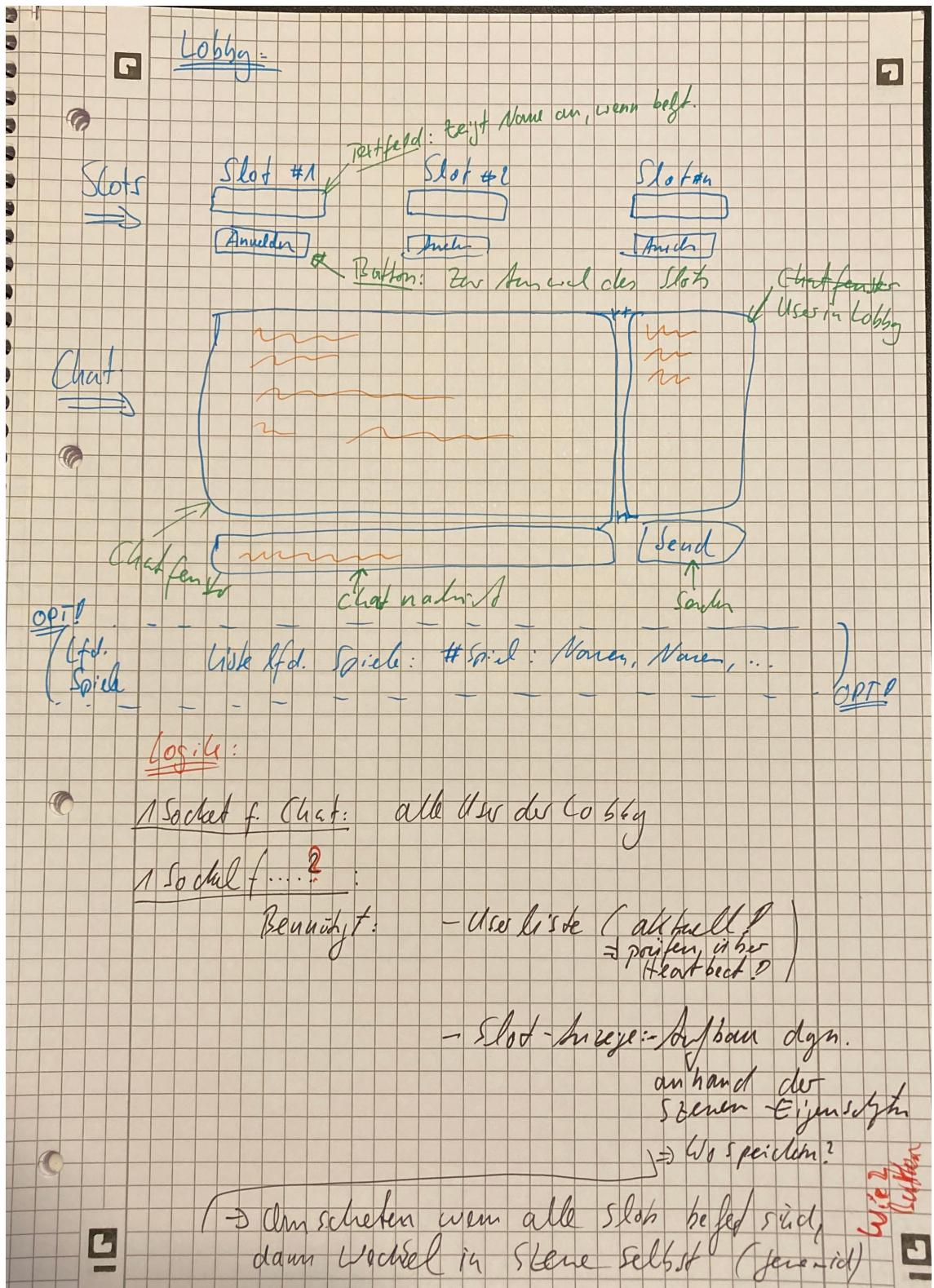
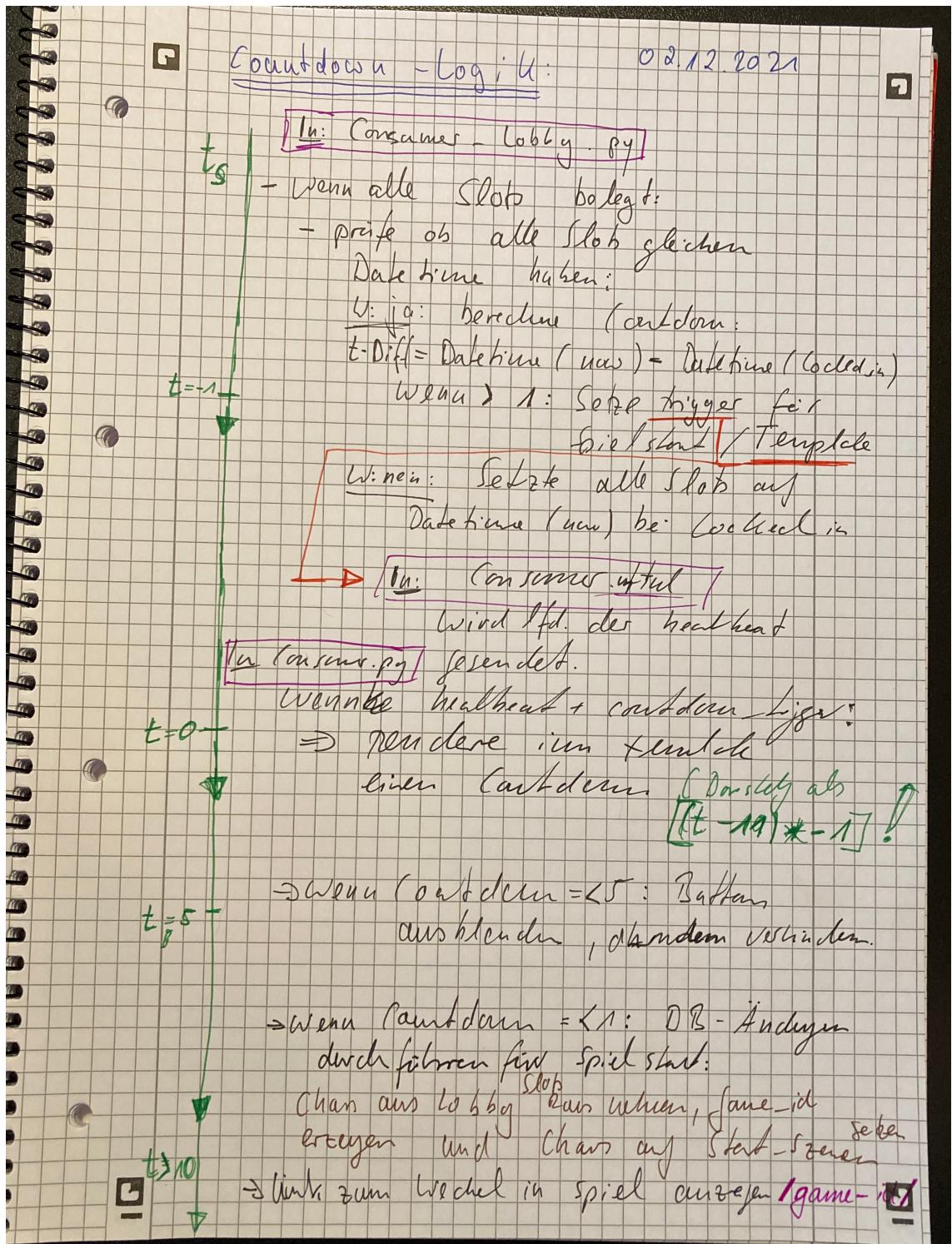


Abbildung 21: Anhang: Konzept zum Zeitstrahl des Countdowns in der Lobby-Logik (02.12.2021)



**Abbildung 22: Anhang: Dokumentation einer Projektbesprechung (05.12.2021)**

**Projekt-Besprechung 05.12.2021:**

- Noch mal aufsetzen der lokalen Entwicklungsumgebungen
- Durchgehen der individuellen Anforderungen für weitere Arbeit:
  - Rico: Feedback zu Ansätzen der Story und dem Setting
  - Julian: Bittet um Konkretisierung der nötigen Grafik-Assets
  - Henning: Kurzes Brainstorming für Input zur Szenen-Logik
- Durchsprechen von Detailanforderungen zu Szenen und dem Ablauf von Spielen:
  - Würfel sollen Teil der Spieldynamik sein
  - Intro-Texte zu jeder Szene (!)
- Gemeinsam Mockup vom Spiel-Screen gezeichnet (s.unten)
- Spiel im Vollbild (ohne Nav usw.)
- Es soll immer nur ein einzelnes Spiel pro User laufen dürfen
- Nächste Programmierungen Henning:
  - Game Scenen erweitern
  - Anzeige der UserChars im Spiel
  - Aufbau der möglichen Aktionen anhand der 'Possible Actions'
  -

To-do's Rico:

- Minimumwerte der Charaktere festlegen
- Klassen ausarbeiten
  - Infotexte etc.
- wie am Besten Kampfmechanik mit Würfelwurf abbilden

- To-Do's Julian:

- Szene schreiben
  - d.h. "Story"-Konzept ausarbeiten
  - Rätsel / Aktionen und generell den Ablauf festlegen
  - Entsprechende Grafiken erstellen (Szenenbilder)

Agenda für nächsten Sonntag (12.12.21):

**Abbildung 23: Anhang: Projekt Besprechung mit gegenseitigem Update und Wechsel von Szenenlogik zu Kampfsystem für das RPG (11.12.2021)**

**Projekt-Besprechung 12.12.2021:**

- Vorstellung erster Zeichnungen und Ideen zu Stats und Charakteren
- Gespräch über Dateiformate, Raster- und Vektorgrafiken.
- SVG Export und Inkscape getestet

**- Ansatz verändert: Weg von Szenenlogik hin zu einem reinen Kampfsystem:**

**Kampfablauf:**

- Begrüßungstext ausgeben ("Hoho, Ihr bekommt Prinzessin Peach nie!")
- Kampf läuft = true
- While(Kampf läuft):
  - Gegner fügt Schaden zu
  - für jeden (lebendigen) Spieler einzeln (in Reihenfolge der Slots):
    - Auswahl Aktion bestimmen (immer 3 Optionen: Schaden machen, Fähigkeit nutzen, Aussetzen [Aggro abbauen])
      - Aktion für Runde speichern
      - Aktionen durchgeführt (div. Rechnungen vorgenommen)
      - Gamelog wird fortgeschrieben
    - Aggrotabelle fortschreiben (Schaden 1:1 Aggro, Verspotten +100 Aggro)
    - stetige Prüfung ob ein HP unter 0 fällt:
      - Wenn Gegner unter 0: Abbruch: Win
      - Wenn kein Spieler mehr am leben ist: Game Over

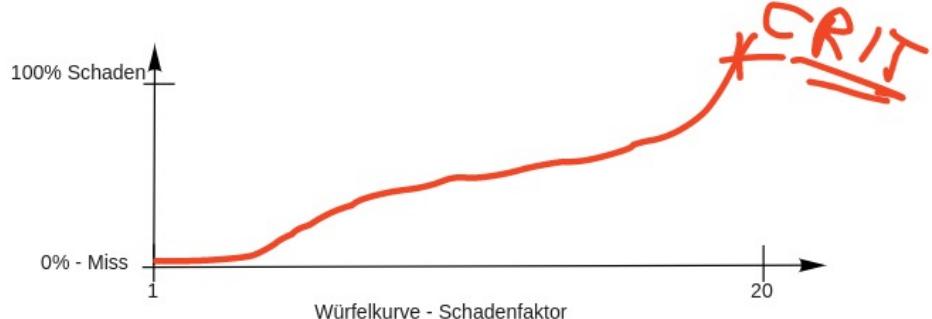
	HP	DMG	Fähigkeit
Gegner:	1500	80-120	
Krieger:	500	20-40	Verspotten (Aggro aufbauen, klingt über Runden ab)
Zauberer:	200	60-80	Buffen (Schaden der Gruppe für x Runden erhöhen)
Priester:	300	30-60	Heilen (HP der Gruppe regenerieren [auch über Runden?])

Levelfortschritt nach Abschluss eines Levels:

- Gegner erledigt?
- ja: HP + 5-10 %, DMG min +8%, DMG max +10%
- nein: nix (kein Game-Progress)

Backlog:

- Mehr Klassen mit anderen Fähigkeiten (Totstellen, Verblissen, ...)



---

**Abbildung 24: Anhang: Projekt Besprechung mit Ermittlung von restlichen ToDos, Aufgabenverteilung, Meilenstein- und Terminplanung (06.01.2022)**

To-Do's:

- Weitere Szenen: 5
  - 2 Multiplayer, 3 Solo
  - 1 Endszene (Endboss)
  - 1 Einstiegsszene
  - 3 Midgame-Szenen
- Inhalte:
  - Grafiken (Szenenbild + Boss)
  - Texte (Intro + Start- / Endtext für jede Szene)
  - Popup bei Szenenstart (Intro), Popup bei Szenenende (Outro)
- Name des Spiels (Setting)
- Synopsis
- Frontend (CSS-Verschönerung)
- Balancing (Werte berechnen (Angriffskraft, Leben, Boss-Angriffskraft, Fähigkeiten, EXP-Verteilung))
- Doku schreiben

Meilensteine:

Setting festlegen: 08.01.2022 (09:30 Uhr) 09.01.2022 (17:00 Uhr)

-> Grafiken erstellen

-> Balancing

-> Texte erstellen 23.01.2022

--> Zusammenbauen: 26.02.2022

Gliederung der Dokumentation festlegen: 09.01.2022

Frontend verschönern (einheitliches Design): 06.02.2022

Dokumentation schreiben

-> Zusammenbauen + Einleitung schreiben: 30.01.2022

## Literaturverzeichnis

heise online, Matthias Wessendorf (2011). *WebSocket: Annäherung an Echtzeit im Web.*  
Verfügbar unter <https://heise.de/-1260189> [15. 06. 2011].

Scheurer, S., Bea, F. X. & Hesselmann, S. (2020). *Projektmanagement*. UTB GmbH.

---

## Ehrenwörtliche Erklärung

Hiermit versichere ich, dass die vorliegende Arbeit von mir selbstständig und ohne unerlaubte Hilfe angefertigt worden ist, insbesondere dass ich alle Stellen, die wörtlich oder annähernd wörtlich aus Veröffentlichungen entnommen sind, durch Zitate als solche gekennzeichnet habe. Weiterhin erkläre ich, dass die Arbeit in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde/Prüfungsstelle vorgelegen hat. Ich erkläre mich damit **nicht einverstanden**, dass die Arbeit der Öffentlichkeit zugänglich gemacht wird. Ich erkläre mich damit einverstanden, dass die Digitalversion dieser Arbeit zwecks Plagiatsprüfung auf die Server externer Anbieter hochgeladen werden darf. Die Plagiatsprüfung stellt keine Zurverfügungstellung für die Öffentlichkeit dar.

---

Siegen, 13.2.2022

(Ort, Datum)

---



(Rico )

---



(Henning )

---



(Julian )