

Bachlorarbeit HS2021

Bachelor of Science - Informatik

Online-Charaktererstellung für «Das Schwarze Auge»

Autor: Mischa Kälin

Betreuungsperson: Dr. Martin Bättig

Experte: Roman Bachmann

Bachelorarbeit an der Hochschule Luzern – Informatik

| Titel: Online-Charaktererstellung für «Das Schwarze Auge» |
|---|
| Student: Mischa Kälin |
| Studiengang: BSc Informatik |
| Abschlussjahr: 2022 |
| Betreuungsperson: Martin Bättig |
| Expertin/Experte: Roman Bachmann |
| Auftraggeber: Mischa Kälin (Smart-Up) |
| |
| Codierung / Klassifizierung der Arbeit: |
| ⊠ Öffentlich (Normalfall) |
| ☐ Vertraulich |
| |
| Eidesstattliche Erklärung |
| Ich erkläre hiermit, dass ich/wir die vorliegende Arbeit selbständig und ohne unerlaubte fremde Hilfe angefertigt haben, alle verwendeten Quellen, Literatur und andere Hilfsmittel angegeben haben, wörtlich oder inhaltlich entnommene Stellen als solche kenntlich gemacht haben, das Vertraulichkeitsinteresse der Auftraggeberin wahren und die Urheberrechtsbestimmungen der Hochschule Luzern respektieren werden. |
| Ort / Datum, Unterschrift: |

Abgabe der Arbeit auf der Portfolio Datenbank:

Bestätigungsvisum Studentin/Student

Ich bestätige, dass ich die Bachelorarbeit korrekt gemäss Merkblatt auf der Portfolio Datenbank abgelegt habe. Die Verantwortlichkeit sowie die Berechtigungen habe ich abgegeben, so dass ich keine Änderungen mehr vornehmen kann oder weitere Dateien hochladen kann.

| Ort / | Datum, | Unterschrift: | |
|-------|--------|---------------|--|
| Ort / | Datum. | Unterschrift: | |

Verdankung

An dieser Stelle richte ich meinen Dank an meine Spielgruppe, die mich mit nützlichem Feedback zu meiner Applikation unterstützt haben. Ausserdem gebührt Christof Kälin mein Dank, der mir seinen privaten Server für die öffentliche Bereitstellung meiner Applikation zur Verfügung gestellt hat.

Ausschliesslich bei Abgabe in gedruckter Form:

Eingangsvisum durch das Sekretariat auszufüllen

|--|

Hinweis: Die Bachelorarbeit wurde von keinem Dozierenden nachbearbeitet. Veröffentlichungen (auch auszugsweise) sind ohne das Einverständnis der Studiengangleitung der Hochschule Luzern – Informatik nicht erlaubt.

Copyright © 2022 Hochschule Luzern – Informatik

Alle Rechte vorbehalten. Kein Teil dieser Arbeit darf ohne die schriftliche Genehmigung der Studiengangleitung der Hochschule Luzern – Informatik in irgendeiner Form reproduziert oder in eine von Maschinen verwendete Sprache übertragen werden.

Abstract

Digitale Hilfsmittel wie ein Charakterbogen gehören zum Repertoire eines modernen Spielers von «Das Schwarze Auge»; denn die Charakterverwaltung hat in den letzten Jahrzehnten an Komplexität und Umfang gewonnen. Eine progressive Web-Applikation, basierend auf Blazor WebAssembly, soll hierfür Abhilfe schaffen und bestehende Lösungen der Fangemeinschaft durch Qualität und Funktionalität übertreffen. Der Fokus dieser Arbeit richtet sich auf die Entwicklung besagter Applikation als Proof-of-Concept. Die Grundlage dazu bilden die Anforderungen aus einem Geschäftsmodell, die anfangs adoptiert und im Verlauf des Projekts durch ein Alpha-Testing, einen Audit und einen Konkurrenzvergleich präzisiert werden. Nicht zuletzt werden technische Verbesserungsvorschläge und eine identifizierte Marktnische aufgezeigt, um die Weiterentwicklung der Applikation zu unterstützen.

Inhaltsverzeichnis

| 1 E | Einleitung | 1 |
|-----|---|----|
| 2 S | Status Quo | 2 |
| 2.1 | Competitive Landscape | 2 |
| 2.2 | Technologien | 4 |
| 3 I | deen und Konzepte | 6 |
| 3.1 | Charakter in DSA | |
| 3.2 | Anwendungsbereich | 7 |
| 4 N | Methodik | 8 |
| 4.1 | Versionskontrolle | |
| 4.2 | Projekt- und Vorgehensmodell | 8 |
| 4.3 | Testkonzept | |
| 5 R | Realisierung | 11 |
| 5.1 | Rechtliche Aspekte & Vertrieb | 11 |
| 5.2 | Anforderungen | 12 |
| 5.3 | Technologieevaluation | 13 |
| 5.4 | Roadblocks | 22 |
| 6 E | Evaluation und Validation | 28 |
| 6.1 | Alpha-Testing | 28 |
| 6.2 | Validation durch Testkonzept | 30 |
| 6.3 | Konkurrenzvergleich | 32 |
| 7 A | Ausblick | 33 |
| 7.1 | Weiteres Vorgehen | 33 |
| 7.2 | Reflexion | 34 |
| 8 A | Anhänge | 36 |
| 8.1 | Lean Canvas | |
| 8.2 | Aufgabenstellung | 37 |
| 8.3 | Projektmanagement | 40 |
| 8.4 | Testdrehbücher | 46 |
| 8.5 | Inhaltsrichtlinie für Scriptorium Aventuris | 47 |
| 8.6 | Anforderungen | 49 |
| 8.7 | Fragebogen | 52 |
| 8.8 | Google Lighthouse: Verbesserungsvorschläge | 55 |

| 9 | Ver | zeichnisse | .58 |
|---|-----|-----------------------|-----|
| | | Abkürzungsverzeichnis | |
| | | Abbildungsverzeichnis | |
| | 9.3 | Codeverzeichnis | 59 |
| | 9.4 | Tabellenverzeichnis | 60 |
| | 9.5 | Literaturverzeichnis | 60 |

1 Einleitung

Moderne Role Playing Games (kurz RPG), wie man sie heute kennt, haben ihre Wurzeln in Dungeon and Dragons (kurz D&D) aus dem Jahr 1974. Dave Arneson und Gary Gygax etablierten mit ihrem Spiel bereits erste Konventionen, die noch bis heute Anwendung finden: Lebenspunkte, Fähigkeiten, Rassen und weitere Charaktereigenschaften sind nur ein Bruchteil davon. Diese Eigenschaften werden klassischerweise auf einem Charakterbogen notiert und während des Spielverlaufs angepasst. Daher stammt der Begriff Pen-&-Paper-RPG, weil lediglich Stift und Papier zum Spielen dieses RPGs benötigt werden. (Beresford 2011)

Seit seiner Veröffentlichung hat sich D&D weiterentwickelt und inzwischen sogar mit über 40 Millionen Spielern weltweit den Mainstream-Status erreicht (Lai and Chen 2020). Dabei prägt die Digitalisierung die Art und Weise, wie Pen-&-Paper-RPGs heute gespielt werden: Der Charakterbogen wird vermehrt digital bearbeitet, während Stift und Papier allmählich in den Hintergrund rücken. D&D hat diesen Trend erkannt und stellt selbst einen digitalen Charakterbogen kostenlos zur Verfügung (Bains 2021).

Dem Erfolg von D&D folgten auch weitere Pen-&-Paper-RPGs. Eines davon ist Das Schwarze Auge (kurz DSA) von Ulrich Kiesow aus dem Jahr 1984. Die aktuelle Edition (DSA5) wurde 2015 veröffentlicht (Schönleben 2015). Im Gegensatz zu D&D bietet DSA aber keine offizielle, digitale Spielhilfe in Form eines Charakterbogens an. Offenbar besteht jedoch hierfür eine Nachfrage in der Community: Aus Eigeninitiative haben vereinzelte Spieler selbst einen Charakterbogen entwickelt und bieten diesen online an (siehe 2.1 Competitive Landscape).

Diese Charaktereditoren weisen jedoch Defizite auf, die im Lean Canvas (siehe 8.1) unter Problemstellung aufgelistet sind: Teilweise wird eine Installation benötigt, die Berechnungen sind fehleranfällig, die Inhalte sind unvollständig und ohne Möglichkeit, eine manuelle Erweiterung vorzunehmen. Als Ausgangslage für die Aufgabenstellung dieser Arbeit (siehe 8.2) dient das festgelegte Geschäftsmodell für einen eigenen Charakterbogen (siehe 8.1 Lean Canvas). Das Ziel ist die Entwicklung einer vollständigen Web-Applikation von den Anforderungen über Konzepte bis hin zur Realisierung unter Berücksichtigung der Techniken der modernen Softwareentwicklung.

2 Status Quo

Innerhalb dieses Kapitels wird zuerst auf das Wettbewerbsumfeld eingegangen. Dabei werden vier Produkte vorgestellt, die als direkte Konkurrenten dieses Projekts betrachtet werden können. In einem zweiten Schritt wird der Stand der Technologien erläutert, welche die Ausgangslage dieses Projekts darstellen.

2.1 Competitive Landscape

Der Bedarf nach digitalen Hilfsmitteln zur Erstellung und Manipulation von Charakteren für DSA wurde bereits von anderen erkannt. Über Ulisses, den Publisher von DSA, werden Charakterbögen aus unterschiedlichen Quellen angeboten. In diesem Teil soll auf vier dieser konkurrierenden Applikationen, die ich selbst bereits in den vergangenen Jahren zur Hilfe genommen habe, eingegangen werden.

Zwei dieser vier Applikationen werden von Ulisses selbst vertrieben. Dazu gehört erstens das DSA5 Deluxe Heldendokument, ein selbstrechnendes PDF mit Ausfüllfeldern, wofür Spieler einen Einmalbetrag entrichten müssen (Ulisses Spiele GmbH 2021). Die Abbildung 1 zeigt einen beispielhaften Ausschnitt aus dem Dokument.

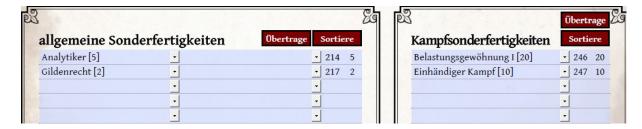


Abbildung 1: Screenshot aus dem DSA5 Deluxe Heldendokument

Die zweite Applikation namens Foundry VTT ist eine Web-Applikation, die nach dem Kauf selbst von Spielgruppen gehostet wird. Der Fokus liegt hierbei auf dem digitalen Spielerlebnis als Gruppe und nicht der Charaktererstellung und -manipulation, obwohl die Applikation dies ermöglicht (Foundry VTT 2021). In der Abbildung 2 ist ein entsprechender Ausschnitt aus dem Frontend von Foundry VTT zu sehen.



Abbildung 2: Screenshot aus Foundry VTT

Die anderen beiden Applikationen heben sich dadurch ab, dass sie eine Installation erfordern. Sowohl der Optolith Character Generator (Obermann 2021) als auch The Dark Aid (Jung 2021) bieten über Ulisses kostenlose Versionen für Linux, macOS und Windows an. Trotz dieser Angaben hatte die Installation von The Dark Aid auf macOS noch visuelle Bugs, die unter Windows nicht aufgetreten sind. Auf der Abbildung 3 ist ein Ausschnitt aus dem Optolith Character Generator dargestellt.

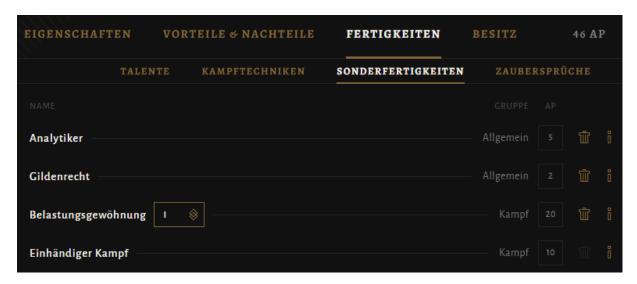


Abbildung 3: Screenshot aus dem Optolith Character Generator

Analog dazu zeigt die Abbildung 4 dieselben Informationen in The Dark Aid. Dabei sind Parallelen in der Darstellung zu erkennen: Beide weisen nebst der Eigenschaftsbezeichnung auch die Kosten aus. Zusätzliche Aktionen wie "Entfernen" und "Mehr Informationen" sind ebenfalls bei beiden vorhanden.

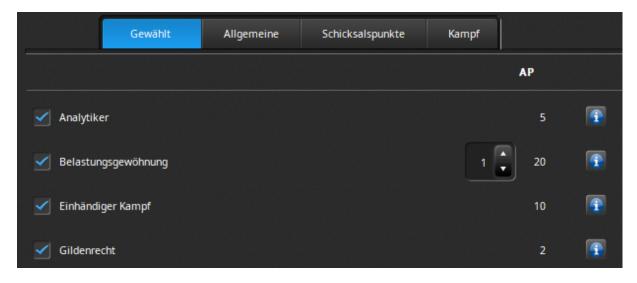


Abbildung 4: Screenshot aus The Dark Aid

Zu den genannten Applikationen existieren weitere Alternativen. Diese finden jedoch bei den Spielern nur selten Anwendung. Oftmals sind fehlende Inhalte und fehlerhafte Berechnungen die ausschlaggebenden Probleme. In seltenen Fällen ist auch die Integrität der Daten nicht gegeben, wodurch Fehler beim Speicherund Ladevorgang aufgetreten können.

2.2 Technologien

Innerhalb dieses Kapitels wird auf die verwendeten Technologien eingegangen. Im Fokus stehen dabei die fortschrittlichen Ansätze und inwiefern diese Vorteile gegenüber traditionellen aufweisen.

2.2.1 WebAssembly

Bei WebAssembly handelt es sich um einen Bytecode, der parallel zu JavaScript im Webbrowser verwendet werden kann. Die Motivation dieser Technologie ist die Beseitigung der Unzulänglichkeiten von JavaScript. Mitunter gehören dazu die unzureichende Performanz sowie die zu weiche Syntax. Letzteres wurde bis anhin versucht durch Supersets wie TypeScript zu lösen, indem ein Compiler dazu gezogen und das kompilierte Resultat zu JavaScript transkribiert wird. Nichtsdestotrotz bleibt das Manko an Leistung bestehen, weil schlussendlich wieder JavaScript ausgeführt wird. (Augsten 2020)

Momentan unterstützen noch nicht alle Browser WebAssembly, wobei sich dies noch ändern dürfte. Das World Wide Web Consortium (kurz W3C) hat nämlich im Jahr 2019 WebAssembly als offiziellen Standard für das Web definiert. Demnach gehört WebAssembly zusammen mit HTML, CSS und JavaScript zu den Sprachen, die jeder Browser nach W3C interpretieren kann. (Couriol 2019)

Weiterhin birgt spezifisch Blazor WebAssembly noch einen weiteren Nachteil in Bezug auf die initiale Downloadgrösse. Obwohl Microsoft stetig daran arbeitet, wird es wahrscheinlich immer eine kompaktere Alternative geben (Vogel 2020). Aufgrund der Anforderungen an dieses Projekt (siehe 5.2) und die daraus resultierende Caching-Strategie (siehe 5.3.3) verliert diese Schwachstelle zunehmend an Bedeutung.

Im Gegenzug profitiert (Blazor) WebAssembly von einer zeitgemässen Implementation in Bezug auf die Sicherheit. Dieser Aspekt hat auch Mozilla dazu bewegt, bereits bestehende Komponenten von Firefox auf WebAssembly zu migrieren. (Froyd 2020)

Die Entscheidung, spezifisch Blazor WebAssembly in diesem Projekt zu verwenden, beruht schliesslich auch auf persönlichen Gründen: Meine bisherige Erfahrung mit und Vorliebe für die Sprache C# in Kombination mit einer modernen und standardisierten Art der Webentwicklung bildet ein stabiles Fundament für die Realisierung dieser Projektidee zu sein. Zusätzlich erlaubt mir dieser Ansatz, für meine zukünftige Karriere einen Einblick in die Welt von WebAssembly zu erlangen.

2.2.2 Progressive Web-Applikation

Eine Progressive Web-Applikation (kurz PWA) ist eine Kombination aus Web-Applikation und nativen Applikation: Sie besteht ausschliesslich aus Web-Technologien (HTML, CSS, JavaScript, WebAssembly), hat aber Zugriff auf Funktionalitäten, die bisher nur nativen Applikationen vorbehalten waren. Darunter fällt mitunter der Zugriff auf Hardware wie die Kamera. Infolgedessen sind PWAs plattformunabhängig und bieten, solange ein Browser vorhanden ist, denselben Funktionsumfang wie native Applikationen – es besteht sogar die Möglichkeit einer Installation einer PWA. (Toonen 2020)

Für die technische Umsetzung einer PWA werden drei Aspekte vorausgesetzt: Erstens muss die Applikation über HTTPS bereitgestellt werden. Diese Anforderung wird dadurch begründet, dass eine PWA potenziell mehr Schaden auf dem System anrichten kann, weil sie Zugriffsrechte ähnlich zu einer nativen Applikation aufweist. Da sich jedoch eine Bereitstellung über HTTPS bereits bei herkömmlichen Web-Applikationen bewährt hat, ist der Zertifizierungsprozess mit wenig Aufwand verbunden. (Morinigo 2019)

Weiterhin wird eine Manifest-Datei in Form eines Json benötigt. Die enthaltenen Meta-Informationen werden verwendet, um die PWA wie eine native Applikation aussehen zu lassen. Dementsprechend werden Name, URL, Icons (Pfad dazu) und weitere Details innerhalb der Manifest-Datei abgelegt (Morinigo 2019). Der Code 1 zeigt beispielhaft das verwendete Manifest-json dieses Projekts.

Code 1: Manifest.json

```
01
    {
02
           "name": "Charaktereditor: Das schwarze Auge",
           "short_name": "rubiQQ",
93
           "start_url": "./",
94
           "display": "standalone",
05
           "background color": "#ffffff",
06
           "theme color": "#03173d",
07
98
           "icons": [
09
                  {
"src": "rubiQQ_512.png",
10
                  "type": "image/png",
11
                  "sizes": "512x512"
12
13
14
           ]
15 }
```

Zuletzt wird ein sogenannter Service-Worker vorausgesetzt (Morinigo 2019). Aus technischer Sicht ist ein Service-Worker ein Script, geschrieben in JavaScript, das Zugriff auf Netzwerkanfragen hat und diese manipulieren kann. Als virtueller Proxy kann der Service-Worker dann Anfragen des Clients abgefangen und beispielsweise auf cached Ressourcen im Browser umleiten. Folglich ergibt sich daraus das Potenzial, zumindest Teile der Web-Applikation nach erstmaliger Benützung auch offline bereitzustellen (MDN Web Docs 2021).

Die beschriebene Anwendungsmöglichkeit ist für dieses Projekt insofern relevant, als dass Anwender nicht vom Inhalt einer spezifischen Version abhängig sind, sondern noch immer Updates in Form von neuen Inhalten erhalten können. Gleichzeitig besteht die Möglichkeit, mit den zuletzt geladenen Inhalten einen Charakter zu erstellen und anzupassen, ohne dabei auf eine stabile Internetverbindung angewiesen zu sein. Letzteres ist eine funktionale Anforderung (siehe 5.2.1) an die Web-Applikation.

Im Gegensatz zu Blazor WebAssembly ist die Entscheidung, eine PWA zu entwickeln, grösstenteils aus den funktionalen Anforderungen (siehe 5.2.1) entsprungen. Die beiden funktionalen Anforderungen an das Produkt, plattformunabhängig (siehe 8.6.4) und gleichzeitig offline lauffähig (siehe 8.6.5) zu sein, lassen kaum Alternativen zu.

3 Ideen und Konzepte

Die Applikation soll weitestgehend den Prozess der Charaktererstellung und -manipulation in DSA unterstützen. Durch eine intuitive Bedienbarkeit, manuelle Erweiterbarkeit, plattformunabhängige Nutzung sowie kurze Ladezeiten werden Vorteile gegenüber der Konkurrenz geschaffen (siehe 8.1 Lean Canvas).

3.1 Charakter in DSA

Für die Entwicklung eines eigenen Charakterbogen muss zunächst geklärt werden, welche Bestandteile und Funktionalitäten dafür notwendig sind. Die Abbildung 5 stellt den Grundaufbau eines Charakters in DSA sowie dessen Eigenschaften dar – sie dient lediglich zur Veranschaulichung der Komplexität und zeigt nicht den technischen Stand des Endprodukts auf.

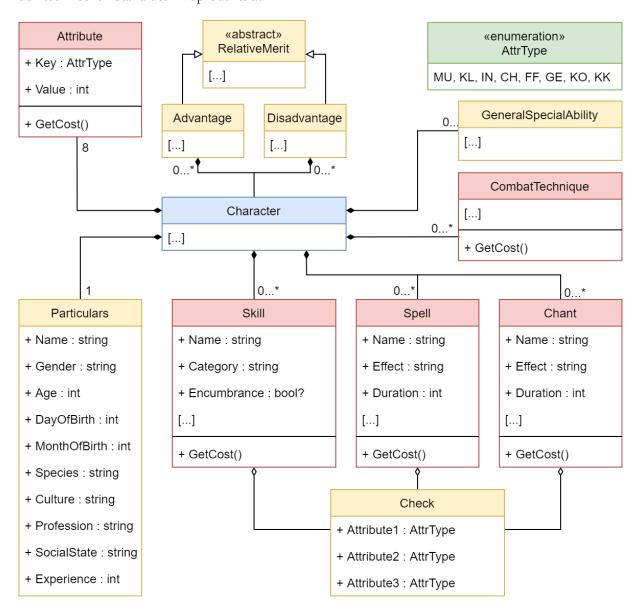


Abbildung 5: Domänenmodell für Charakter in DSA (Grundregelwerk)

Das Modell in Abbildung 5 beruht auf den Regeln des Grundregelwerks. Zusätzliche Erweiterungspakete können eine Anpassung des Modells verursachen, indem weitere Attribute oder Abhängigkeiten eingeführt werden. Ziel ist die Verwendung dieses Datenmodells bei Client und Server.

3.2 Anwendungsbereich

Das Produkt dieses Projekts dient als Hilfsmittel für die Charaktererstellung bei Spielbeginn und die Charaktermanipulation im Verlauf des Abenteuers, wenn sich der Charakter weiterentwickelt. Auf dieser Voraussetzung eines sich entwickelnden Charakters (Character Development) basiert das Spielprinzip von Pen-&-Paper-RPGs: Die Spieler nehmen die Rolle eines fiktiven Charakters ein und bestreiten mit diesem meist über mehrere Spielsessions hinweg ein vorskizziertes Abenteuer.

Die einzige Ausnahme hierfür bildet der Gamemaster, welcher keinen eigenen Charakter besitzt. Diese Person ist für den Erzählfluss, die Durchsetzung der Regeln sowie die Umwelt zuständig (Tychsen, et al. 2015). Darunter fallen unter anderem Interaktionen zwischen Mitspieler und sogenannten Non-Player Characters (kurz NPC). Obwohl diese NPC auch Charaktere sind, lohnt es sich kaum, eine Repräsentation dieser im Charakterbogen zu erstellen. Grund dafür ist der bescheidene Aufgabenbereich von NPCs und die entsprechend geringe Charaktertiefe.

3.2.1 Erfahrungsgrad

Trotz der Einfachheit und Bedienbarkeit richtet sich das Produkt aber nicht nur an neue Spieler, die noch wenig Erfahrung mit DSA haben. Das Produkt soll eine übersichtliche Struktur mit einer manuellen Erweiterbarkeit kombinieren, sodass auch erfahrene Spieler davon profitieren.

3.2.2 Plattform

Trotz der Namensgebung Pen-&-Paper findet heutzutage mindestens die Charaktererstellung auch digital statt. Digital bedeutet aber nicht zwingend online. Komplexe Regeln und eine Vielzahl an Berechnungen sind bei manueller Ausführung anfälliger für Fehler. Bei weiterführenden Charaktermanipulationen ist dies weniger schwieriger, da lediglich inkrementell Änderungen stattfinden. Die Verwendung ohne ausreichende Internetverbindung ist ein Anwendungsfall, auf den später vertieft eingegangen wird.

Weiterhin werden spätestens seit der COVID-19-Pandemie vermehrt digitale Hilfsmittel von Spielern herangezogen, weil zu dieser Zeit digitale Spielrunden oftmals die einzige Option waren (Haslhofer 2020). Aber bereits davor hat sich ein Grossteil der Spieler, insbesondere junge, vom traditionellen Papier und Stift gelöst und entweder auf ein Tablet oder einen Laptop gewechselt. Aufgrund dessen ist die Bedienoberfläche des Produkts auch auf diese beiden Mediengrössen optimiert, obwohl ein Mobile First Ansatz in den meisten Fällen zeitgerechter erscheinen mag (Schwarz and Grote 2020).

4 Methodik

Abgesehen von den definierten Technologien (WebAssembly und PWA) werden technische Entscheide während der Implementation durch Recherchen, Gegenüberstellungen sowie persönlichen Erfahrungen getroffen. Auf solche Evaluationen und weitere technische Hindernisse wird detailliert in der Realisierung (siehe 5) eingegangen.

Das Produkt dieser Arbeit dient als Fundament, um erstes Feedback von potenziellen Anwendern zu sammeln. Aus diesem Grund wird erst innerhalb der Evaluation und Validation (siehe 6) direkt auf die Erwartungen der persönlichen Spielgruppe, bestehend aus fünf weiteren Personen, eingegangen. Grund dafür ist mitunter die benötigte Expertise für eine zielführende Diskussion von präzisen Anforderungen zum Startzeitpunkt. Die Aufbereitung der Anforderungen (siehe 5.2) aus dem Lean Canvas (siehe 8.1) enthält bereits die kritischen Punkte zur Benützung der Software, die aus eigenen Erfahrungen stammen.

4.1 Versionskontrolle

Innerhalb dieses Projekts wird Git zur Versionsverwaltung des Codes verwendet. GitLab ist dabei der Host für das Repository, was die Verknüpfung mit DevOps-Prozessen wie Continuous Integration (kurz CI) bereits anbietet. Gleichzeitig wird das Backlog zusammen mit den Sprints auf GitLab aktiv geführt.

4.2 Projekt- und Vorgehensmodell

Als Vorgehensmodell für Lehre, Forschung und Industrie in der Informatik bietet sich SoDa (siehe 9.1 Abkürzungsverzeichnis) auch für diese Arbeit an (Jud, et al. 2020). Aufgrund der agilen Natur von SoDa ist die Reihenfolge von Aufgaben innerhalb der Sprints spontan vertauschbar. Dadurch ergibt sich eine gewisse Freiheit, ohne dabei die übergreifende Zielsetzung aus den Augen zu verlieren: Vorab definierte Meilensteine helfen dabei, den Fortschritt der Arbeit zu beurteilen. Die Abbildung 6 visualisiert das Vorgehensmodell anhand des Projekts, bestehend aus Initialisierung (gelb) und Realisation (blau).

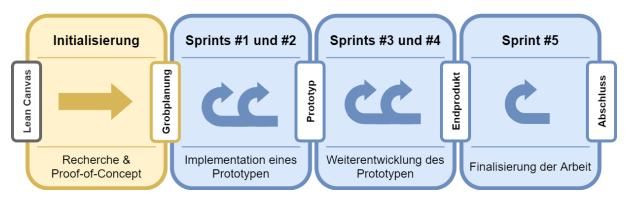


Abbildung 6: Software Development agile

In der Initialisierungsphase wird ein Proof-of-Concept und eine Grobplanung für das Projekts erarbeitet. Dazu gehört die grobe Erfassung der relevanten Arbeitspakete für die ersten zwei Sprints sowie das Setzen von weiteren Sprints und Meilensteinen innerhalb der Projektzeit.

Die gesamte Realisation findet in den fünf agilen Sprints statt, wobei sich die ersten beiden Sprints auf die Fertigstellung eines Prototyps fokussieren. Dieser dient zugleich als Meilenstein wie auch als Prototyp für die Zwischenpräsentation. Am Ende des vierten Sprints ist eine fertige Version geplant, um im fünften Sprint noch die Implementation einer ersten Version als PWA zu realisieren. Zusätzlich zielt der letzte Sprint noch auf Bugfixes ab und enthält die abschliessenden Arbeiten an der Dokumentation.

Analog zu Scrum besteht jeder Sprint aus einer Planung, einer Abnahme sowie einer Retrospektive. Zudem ist auch bei SoDa der inkrementelle Release das Ziel eines jeden Sprints. Die Abbildung 7 zeigt den Vorteil dieser Lösung in Bezug auf unbeständige Anforderungen und frühzeitige Rückmeldungen von Nutzern. Das endgültige Ziel dieses Projekts ist ein stabiles Fundament für eine PWA zu legen, dabei aber trotzdem ein funktionierendes Produkt an Testern schnellstmöglich zur Verfügung zu stellen – wenn auch offline.

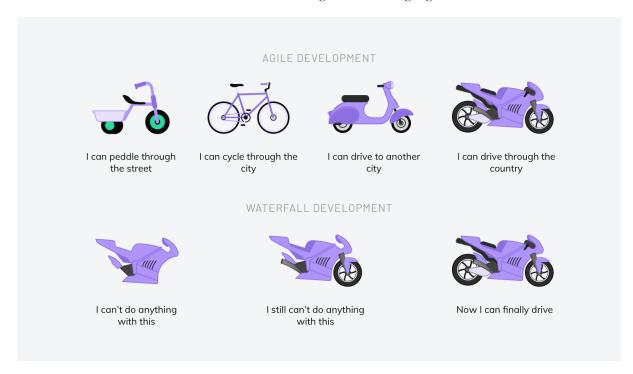


Abbildung 7: Unterschied von Agil und Wasserfall (Amsterdam Standard 2021)

Damit ein gemeinsames Verständnis für den Abschluss von einzelnen Arbeitspaketen besteht, wird die Definition-of-Done (kurz DoD) eingeführt: Eine Menge an Kriterien, die zur Gewährleistung der Qualität erfüllt werden müssen, bevor ein Arbeitspaket als Teil der Applikation veröffentlicht wird.

- Erfüllung der nicht-funktionale Anforderungen
- Integration im Testkonzept (mit Unit-Tests und/oder Integrationstests)
- Fehlerlose Kompilation des Projekts
- Automatisierte Tests erfolgreich
- Dokumentation erweitert (mindestens stichwortartig)

4.3 Testkonzept

Ein umfassendes Testkonzept erhöht die Qualität der Software, indem Programmfehler und Diskrepanzen zu den Anforderungen frühzeitig entdeckt werden. Dabei wird zwischen automatisierten und manuellen Tests unterschieden, welche noch weiter differenziert werden. (Hamilton 2021)

Innerhalb dieses Projekts wird zwischen drei Arten von Tests unterschieden: Unit-Tests, Integrationstests (zur Überprüfung der funktionalen Anforderungen) und Systemtests (End-to-end). Sämtliche Unit-Tests werden automatisiert ausgeführt, um möglichst kleine Module im Code effizient zu überprüfen. Ähnlich wird auch ein Teil der Integrationstests automatisiert ausgeführt, mit dem primären Ziel, die funktionalen Anforderungen (siehe 8.6) vor dem Abschluss des jeweiligen Backlog-Items zu überprüfen. Systemtests werden ausschliesslich manuell und lediglich am Ende jedes Sprints durchgeführt.

4.3.1 Continuous Integration

Als Grundlage für das automatisierte Testing dient die sogenannte Continuous Integration (kurz CI), die von GitLab zur Verfügung gestellt wird. Die benötigte Konfiguration kann direkt beim Erstellen des Repositorys durch GitLab generiert werden. Dadurch wird bei jedem Push sowohl ein Build (Kompilation) wie auch die angegebenen Tests ausgeführt. Für Letzteres wird zusätzlich noch ein Framework benötigt, weshalb in den separierten Test-Projekten Nunit installiert ist.

In erster Linie werden bei der CI die Unit-Tests für das Datenmodell ausgeführt und auf ihre Korrektheit überprüft. Ziel ist es, einen Grossteil der Logik innerhalb des Datenmodells damit zu überprüfen, weil damit die Integrität und Korrektheit des Charakters gewährleistet wird.

Weiterhin werden auch die Blazor-Komponenten durch automatisierte Tests auf Änderungen und/oder Fehler überprüft. Hierbei wird der Fokus auf eine korrekte Anzeige gelegt, indem die jeweilige Komponente mit Parametern erstellt und anschliessend anhand des erwarteten HTML Codes abgeglichen wird. Für die Komponenten mit API-Calls (respektive Zugriff auf wwwroot) wird ein Mock verwendet. Mit dieser Art von Tests soll nebst den Kosten nochmals Wert auf die exakte Anzeige des Charakters gelegt werden.

4.3.2 Manuelle Tests

Für die manuellen Tests werden Testdrehbücher als Schritt-für-Schritt-Anleitungen inklusive Ausgangslage verfasst. Ziel ist die systematische Überprüfung der erwarteten Funktionalitäten mit einer möglichst hohen Abdeckung der Anforderungen. Im Anhang (siehe 8.4 Testdrehbücher) befinden sich die bestehenden Testdrehbücher, die zur Qualitätssicherung fortlaufend hinzugefügt wurden.

Die Testdrehbücher dienen zur sorgfältigen Überprüfung der Änderungen eines jeden Sprints, weshalb diese nicht nach Abschluss jedes Arbeitspaketes durchgeführt werden. Eine zusätzliche Durchführung kann bei grossen Änderungen am Code sinnvoll sein, um allfällige Probleme frühzeitig zu erkennen. Ansonsten kann dies dazu führen, dass sich der Abschluss des Arbeitspakets in den nächsten Sprint verzögert.

5 Realisierung

In diesem Kapitel wird die Initialisierungsphase des Projekts dargelegt, wobei auf rechtliche Aspekte, den Vertrieb und das Requirement-Engineering eingegangen wird. Daraufhin werden kritische Fragestellungen aus der Realisierungsphase gemeinsam mit den entsprechenden Lösungsansätzen erläutert. Das Ziel ist es, wertvolle Erfahrungswerte innerhalb der Software-Entwicklung im Bereich Web wiederzugeben.

5.1 Rechtliche Aspekte & Vertrieb

Der zu erstellende Charakterbogen enthält offenkundig geistiges Eigentum von Ulisses, wodurch eine Erlaubnis dieser unumgänglich ist. Glücklicherweise ist es bei DSA üblich, selbst Abenteuer, Tools oder Kreaturen zu erstellen und zu veröffentlichen. Deshalb hat Ulisses ein Programm namens «Scriptorium Aventuris» geschaffen, welches diese Publikation ermöglicht. Inbegriffen ist auch eine schriftliche Erlaubnis zur Nutzung des geistigen Eigentums, insofern die Inhalte für DSA sind:

«Das Scriptorium Aventuris ist ein Programm, das es dir erlaubt, Inhalte (Abenteuer! Orte! Monster! Und mehr!) für Das Schwarze Auge unter Nutzung des geistigen Eigentums von Ulisses Spiele zu erstellen. Erstelle deine eigenen DSA-Titel entsprechend der Inhaltsrichtlinien des Programms, lade deine Titel in den E-Book-Marktplatz von Ulisses hoch und teile sie mit anderen DSA-Fans – so kannst du etwas verdienen und anderen und dir selbst eine Freude machen.» (Ulisses Spiele GmbH 2021)

Im Anhang (siehe 8.4.4) befinden sich die ausführlichen Richtlinien vom «Scriptorium Aventuris», um eigene Inhalte zu erstellen. Innerhalb dieser Richtlinien befindet sich mitunter eine Liste der erlaubten Medien, die Rollenspielergänzungen (Charakterbogen) beinhaltet. Weiterhin umfassen diese Richtlinien einen Text, welcher im Werk selbst, wo alle rechtlichen und Copyright-Hinweise aufgeführt sind, enthalten sein muss (siehe 8.5.7 Rechtliches). Schliesslich sind noch vereinzelt Verbote enthalten, die sich auf den Inhalt der Texte fokussieren. Darunter fallen rassistische Inhalte, politische Ansichten oder kriminelle Handlungen gegenüber Kindern.

Entsprechend ist der Vertrieb nur über das «Scriptorium Aventuris» möglich, solange keine anderweitigen Abmachungen in Form von Verträgen mit Ulisses geschlossen werden. Darauf kann ein fester Preis oder aber ein vorgeschlagener Preis angegeben werden. Beim Letzteren hat jeder Käufer die Möglichkeit, das Produkt ohne Kosten zu beziehen. Die letzte Alternative ist die Option eines Spendenbuttons, wobei hier wieder die finanzielle Entscheidung dem Käufer überlassen wird.

Für die erste Version auf dem «Scriptorium Aventuris» empfiehlt sich ein unabhängiger Client, der nicht auf die Kommunikation mit einem Server abhängig ist. Grund dafür ist zunächst der weniger riskante Rollout, da der Zugriff zur Applikation durch verkaufte Logins durch das «Scriptorium Aventuris» nicht geregelt ist.

5.2 Anforderungen

Grundsätzlich stammen alle Anforderungen von zwei Quellen ab: Einerseits den rechtlichen Grundlagen und andererseits den persönlichen Erwartungen und Wünschen der Nutzer. Während Ersteres bereits im vorherigen Kapitel besprochen wurde und mehrheitlich die Rahmenbedingungen festlegt, müssen die Anforderungen aus Sicht der Nutzer erst aus dem Lean Canvas (siehe 8.1) adoptiert und allenfalls spezifiziert werden. Durchaus sind Ergänzungen aufgrund von persönlichen Erfahrungen oder Erfahrungen dritter möglich und entsprechend gekennzeichnet.

5.2.1 Funktionale Anforderungen

Beruhend auf dem Lean Canvas (siehe 8.1) und der Aufgabenstellung (siehe 8.2) ergeben sich Schwerpunkte, die sich als Epics – oder im Fall von kleineren Anliegen als Arbeitspakete – im Backlog wiederfinden. Die folgende Aufzählung zeigt die Gesamtheit der funktionalen Anforderungen:

- Charaktererstellung und -manipulation
- Einhaltung aller rechtlichen Aspekte
- Persistenz des Charakters
- Keine lokale Installation notwendig
- Lauffähig offline
- Berechnungen sind korrekt und komplett
- Validierung der Abhängigkeiten (Integrität des Charakters)
- Manuelle Erweiterbarkeit zur Verfügung stellen
- Standardisierter Json-Export als Schnittstelle
- Standardisierter Json-Import als Schnittstelle

Im Anhang (siehe 8.6 Anforderungen) befindet sich eine priorisierte Auflistung der genannten, funktionalen Anforderungen mit zusätzlichen Attributen zur besseren Nachvollziehbarkeit (Rupp 2021).

5.2.2 Nicht-funktionale Anforderungen

Die nicht-funktionalen Anforderungen ergeben sich zum einen aus dem Lean Canvas (siehe 8.1) und zum anderen aus den rechtlichen Aspekten und dem Vertrieb (siehe 5.1). Im Vordergrund stehen hierbei nur schwer messbare Eigenschaften, die direkt vom Nutzer abhängig sind: Übersichtlichkeit, Verständlichkeit, Einfachheit. Aber auch die Performanz der Software ist als nicht-funktionale Anforderung aufgelistet, weil bewusst auf keinen numerischen Grenzwert, sondern auf die Akzeptanz des Anwenders gesetzt wird.

Die Wahl der Technologie, der Software-Architektur und anderen Entscheidungen wird stets anhand dieser Kriterien überprüft, weil sie die Rahmenbedingungen des Projekts beschreiben. Deshalb werden diese nichtfunktionalen Anforderungen innerhalb der DoD (siehe 4.2 Projekt- und Vorgehensmodell) berücksichtigt und bei der Evaluation und Validation (siehe 6) durch die Spielgruppe überprüft.

5.3 Technologieevaluation

Als Ausgangslage für dieses Kapitel gilt die Entscheidung, eine PWA mithilfe von Blazor WebAssembly zu realisieren (siehe 2.2 Technologien). Die Verwendung dieser Technologien beeinflusst zudem die Auswahl weiterer Technologien zur Verwirklichung der Anforderungen.

Im Laufe der Realisation dieses Projekts mussten mehrmals im Kontext von Roadblocks oder Konzepten Entscheidungen getroffen werden. Die einflussreichsten dieser Entscheidungen werden in diesem Kapitel in nicht chronologisch Reihenfolge, sondern nach Themengebiet beschrieben.

5.3.1 Software-Architektur

Eine frühzeitige Entscheidung betreffend Software-Architektur lohnt sich nach Johner (2019). Denn kontinuierliches Refactoring ist nicht nur aufwändig, sondern überfordert oftmals die Kapazität vieler Entwickler wegen des Drucks, gleichzeitig eine Wertschöpfung zu generieren:

«Die eigentliche Entwicklung und Wertschöpfung findet beim Entwerfen der Software-Architektur statt – und (hoffentlich) nicht beim Programmieren. Wer das nicht verstehen will, bezahlt gleich mehrfach» (Johner 2019)

Die möglichen Konsequenzen sind erhöhter Zeitaufwand, Minderung der Produktintegrität sowie eine erschwerte Wartbarkeit aufgrund einer inkonsistenten Architektur (Johner 2019). Um dieser Problematik vorzubeugen, wird anfänglich eine geeignete Software-Architektur definiert.

Aus der Idee, dass voraussichtlich dasselbe Datenmodell für den Charakter im Client und Server verwendet wird, besteht bereits die erste Bedingung an die Software-Architektur: Das Datenmodell muss extrahierbar sein – bestenfalls als NuGet Package für eine möglichst einfache Handhabung.

Aus den funktionalen Anforderungen (siehe 5.2.1) geht zudem hervor, dass die Datenintegrität, spezifisch die Konsistenz und Genauigkeit, eine Anforderung an die Applikation ist. Folglich muss auch die Software-Architektur darauf ausgelegt sein.

Nicht zuletzt ist die Komplexität ein zu beachtender Faktor: Solange keine relevanten Vorteile erlangt werden, soll die Architektur nicht komplexer als notwendig ausfallen. Dies begünstigt unmittelbar eine kürzere Planungsphase gefolgt von einer simpleren Einführungsphase.

Zur Auswahl stehen fünf der weitverbreitetsten Software-Architekturen: Layered (n-tier), Event-driven, Microkernel, Microservice und Space-based. Je nach Anwendungsfall können diese auch in Kombination innerhalb derselben Applikation Verwendung finden. (Wayner 2015)

Schichtenmodell

Als vorherrschende Software-Architektur profitiert die Layered-Architecture von einer übergeordneten Struktur, die als Ausgangslage definiert wird. Beruhend auf den Anforderungen können mehr oder weniger Ebenen definiert werden, wobei drei bis vier Ebenen üblich sind: Präsentation, Applikation, Domäne und Infrastruktur. (Ziemoński 2017)

In diesem Projekt steht die Präsentation für die Benutzeroberfläche, welche durch das UI Framework (siehe 5.3.4) mittels Blazor realisiert wird. Die Applikation beläuft sich auf die logischen Abläufe und ist somit mitverantwortlich für die Umsetzung der funktionalen Anforderungen (siehe 5.2.1). Zur Domäne gehören die Datenmodelle, isoliert von den restlichen Ebenen. Die letzte Schicht bildet die Infrastruktur, zuständig für die Persistenz der Daten, aus einem Speichersystem und dessen Anbindung. (Ziemoński 2017)

Die Technologie Blazor WebAssembly, zusammen mit der Idee eines extrahierbaren Datenmodells, gibt also bereits eine strikte Aufteilung in mindestens vier Schichten vor. Die Abbildung 8 stellt das Modell der geplanten Software-Architektur für dieses Projekt dar.

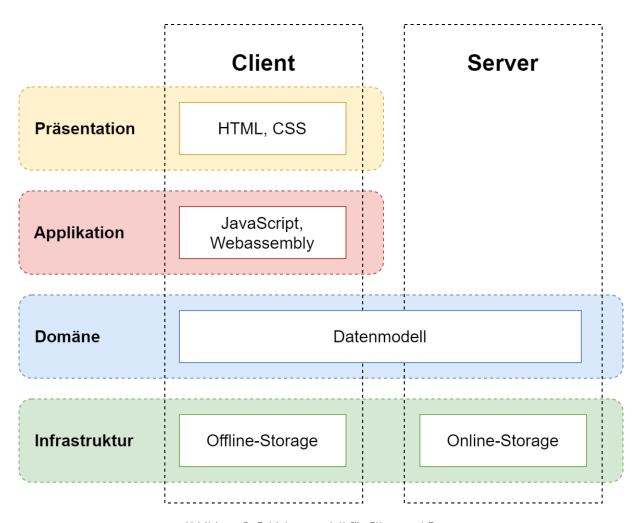


Abbildung 8: Schichtenmodell für Client und Server

Der Offline-Storage wird für den Offline-Modus der PWA benötigt. Die Domäne bildet dabei die Basis der Kommunikation zwischen Client und Server, da beide vom selben Datenmodell ausgehen. Damit jedoch, wie im Projekt- und Vorgehensmodell (siehe 4.2) beschrieben, ein funktionierendes Produkt am Ende jedes Sprints zur Verfügung steht, wird auf die Implementation und Bereitstellung eines Servers verzichtet.

Bei einer zukünftigen Bereitstellung einer API für die Kommunikation zwischen Client und Server muss darauf geachtet werden, dass die Rückwärtskompatibilität gewährleistet ist, solange Benutzer noch ältere Versionen davon benutzen. Dies ist eine Problematik, die sich aus der PWA ergibt, weil die Applikation auch offline und somit ohne Updates lauffähig ist.

Ausserdem werden rechtliche Aspekte in Bezug auf den Vertrieb (siehe 5.1) umgangen, indem vorerst keine Logins verkauft werden, sondern lediglich der Client als komplettes Produkt angeboten wird. Das weitere Vorgehen hängt von der Zusammenarbeit mit Ulisses ab und übersteigt dadurch die Reichweite dieses Projekts. Im Ausblick (siehe 7) wird nochmals detailliert auf diese Problematik eingegangen.

Kombinationsmöglichkeiten

Im geplanten Umfang dieses Projekts (siehe 5.2 Anforderungen) bestehen kaum Vorteile, wenn zusätzliche Architekturansätze mit der Schichtenarchitektur kombiniert werden: Die Verständlichkeit nimmt ab, ohne dabei Schwachstellen auszumerzen. Besonders die beiden Architekturen Event-driven und Space-based wirken sich negativ auf die Nachvollziehbarkeit des Gesamtsystems aus.

Sollte allerdings das Produkt dieser Arbeit zukünftig vermehrt mit Features ergänzt werden, kann sich eine Kombination mit einer Microkernel-Architektur als nützlich erweisen. Der Charakterbogen bildet dabei den Kern. Diese Architektur erweist sich dann als sinnvoll, wenn die Features in Form von austauschbaren Plugins verwendet werden. Obwohl dadurch die Komplexität des Systems zunimmt, wird gleichzeitig die Flexibilität und die Erweiterbarkeit positiv beeinflusst (Morlion 2018).

Diese Vorteile bietet auch die Realisation als Microservice-Architektur. Zudem wird das System mit weniger Aufwand skalierbar und die einzelnen Services können unabhängig ersetzt werden. Falls diese letzten Vorteile in der Praxis nicht als prioritär erachtet werden, wird von dieser Architektur abgeraten. Denn die Schwierigkeiten mit Blick auf die Transparenz bei der Fehlersuche, Kommunikation, Koordination und Rückwärtskompatibilität verlangen ein ausgeklügeltes System. Ansonsten verlieren die Vorteile ihren Wert, weil die Funktionalitäten nicht gewährleistet sind.

Falls eine solche Kombination gewählt wird, bietet sich je nach Features eine Aufteilung des Datenmodells an, um die Verständlichkeit möglichst aufrechtzuerhalten. Ein eigenes Datenmodell pro Microservice oder Plugin mindert die Komplexität und zieht weniger Overhead mit sich – entgegen dem Vorteil eines einzigen Datenmodells wie in Abbildung 8 dargestellt. Beide Optionen ermöglichen die Erweiterung vom Ergebnis dieser Arbeit auf elegante Art und Weise.

5.3.2 Datenpersistenz

Die Persistenz der Daten bezieht sich ausschliesslich auf die Infrastruktur-Ebene aus dem vorhergehenden Kapitel und kann daher in zwei Kategorien unterteilt werden: Client und Server respektive Offline- und Online-Storage.

Offline-Storage (Client)

Die Speicheroptionen im Browser beschränken sich auf die Cache API als Teil des Service-Workers und der IndexedDB. Obwohl es keine strikte Regelung zu den Verwendungszwecken gibt, ist die Cache API für URL adressierbare Ressourcen gedacht. Für PWAs beinhaltet dies alle statischen Daten der App Shell (HTML, CSS, JavaScript, WebAssembly) sowie die unveränderlichen Daten aus dem Datenmodell. Für alle anderen Daten wird die IndexedDB empfohlen. (Osmani 2016)

Weitere Speicheroptionen wie Local Storage, File System API oder WebSQL eignen sich nicht aufgrund ihrer limitierten Grösse und/oder Anbindungsmöglichkeiten – der Local Storage ist beispielsweise für Web und Service-Worker nicht verfügbar. Möglicherweise bietet sich noch die Möglichkeit, den Session Storage für spezifische Daten zur Session zu speichern. Zu beachten ist jedoch der synchrone Zugriff, welcher sich negativ auf die Usability auswirken kann. (LePage 2020)

Im Kontext dieses Projekts, in welchem kein Server implementiert ist, wird die Cache API im Service-Worker verwendet, um die abrufbaren Ressourcen aus dem wwwroot lokal abzuspeichern. Dieser Vorgang wird für die Verwendung ohne Netzwerk benötigt und bei der Caching-Strategie (siehe 5.3.3) detailliert beschrieben.

Online-Storage (Server)

Im Gegensatz zum Client, der von den Funktionen des installierten Browsers abhängig ist, kann der Server unabhängig vom Nutzer und anhand der Anforderungen aufgesetzt werden, solange er für den Client via API zur Verfügung steht. Idealerweise wird wie angedacht dasselbe Datenmodell als NuGet wie beim Client verwendet, wodurch eine Einschränkung bezüglich der Programmiersprache besteht.

Bei der Umstellung der Applikation auf eine PWA erwies sich die Lösung ohne Server jedoch als realistische Option, weil gegenwärtig nebst statischen Daten keine Ressourcen verwendet werden. Diese Alternative wird standalone Blazor WebAssembly genannt und steht im Kontrast zu hosted Blazor WebAssembly (Microsoft 2021). Dieser Ansatz ist aufgrund seiner Einfachheit besonders interessant, indem lediglich eine Applikation realisiert, gepflegt und bereitgestellt werden muss.

Weil der Entscheid bezüglich standalone oder hosted Blazor WebAssembly noch aussteht, wird nicht weiter auf die verschiedenen Ansätze für den Online-Storage eingegangen. Ausgehend von dieser Situation ist möglicherweise ein Fokus auf ein Backend, das auf die Authentifizierung ausgelegt ist, sinnvoller. Allerdings ist der Bedarf hierbei auch noch nicht geklärt.

5.3.3 Caching-Strategie

Wie bereits im Kapitel Progressive Web-Applikation (siehe 2.2.2) erläutert, bietet der Service-Worker ein Caching von Ressourcen an. Dies ist aber nicht in jedem Fall sinnvoll: Je nach Bedarf kann ein Request über das Netzwerk von Vorteil sein, damit aktuelle Informationen für zeitkritische Inhalte vorhanden sind.

In einem ersten Schritt muss also entschieden werden, welche Ressourcen überhaupt im lokalen Cache zwischengespeichert werden. Diese Fragestellung ist mitunter abhängig davon, welche nicht-statischen Daten für die Lauffähigkeit offline kritisch sind. Dazu gehören in erster Linie die Bausteine, die zur Darstellung der PWA benötigt werden: Libraries (WebAssembly), Scripts (JavaScript), HTML, CSS, Bilder Favicon und Manifest. Spezifisch für dieses Projekt kommen noch die applikationsrelevanten Inhalte zur Erstellung und/oder Manipulation des Charakters hinzu.

Nachdem die Frage geklärt ist, was im Cache landen soll, kann erst auf die Caching- oder auch Fetching-Strategie eingegangen werden. Diese beschreibt, wie der Prozess beim Fetch von Ressourcen aussieht, also wann auf den Cache zurückgegriffen und wann das Netzwerk bevorzugt wird. Seit der Einführung des Service-Workers haben sich eine Reihe an Strategien herausgebildet, die sich seither bewährt haben und nachfolgend erläutert werden. (Theo 2021)

Netzwerk oder Cache

Die erste Fetching-Strategie beschreibt eine traditionelle Homepage, bei der jegliche Ressourcen über das Netzwerk geladen werden und kein Caching stattfindet. Die Abbildung 9 stellt dies visuell dar. Network-Only ist somit primär beim Umgang mit sich schnell ändernden Daten sinnvoll, jedoch inkompatibel mit der Anforderung Lauffähig offline (siehe 8.6.5). Deshalb wird nicht weiter auf diese Strategie eingegangen.

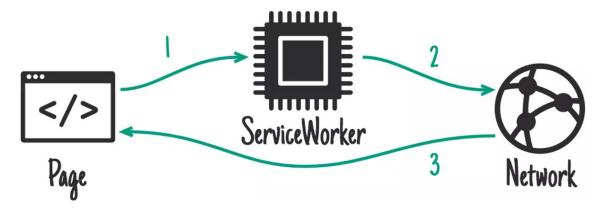


Abbildung 9: Network only (Theo 2021)

Analog zu Network-Only wird bei Cache-Only auf eine Kombination von Netzwerk und Cache verzichtet. In diesem Fall sind keine Requests über das Netzwerk möglich, wodurch die Applikation zu einer reinen offline Seite degeneriert. Diese Strategie erlaubt keine Updates oder Authentifizierung über das Netzwerk, wodurch das Risiko, in Konflikt mit zukünftigen Anforderungen zu stehen, höher als bei den Alternativen ist. Ausserdem bestehen keine Vorteile für dieses Projekt gegenüber den nachfolgenden Strategien.

Kombinationen aus Netzwerk und Cache

Mit dem Fokus auf das Netzwerk aber mit einem Cache als Fallback dient Network-First als ideale Lösung für sich regelmässig ändernde Daten, die aus Synchronisationsgründen wann immer möglich neu über das Netzwerk abgefragt werden sollten. Die Abbildung 10 zeigt die Abfolge an benötigten Aktionen, wobei die Nummer 3 nicht immer eine Abfrage, sondern auch eine Synchronisation des Caches sein kann. Bei jeder erfolgreichen Abfrage wird der Cache mit dem empfangenen Daten aktualisiert. (Theo 2021)

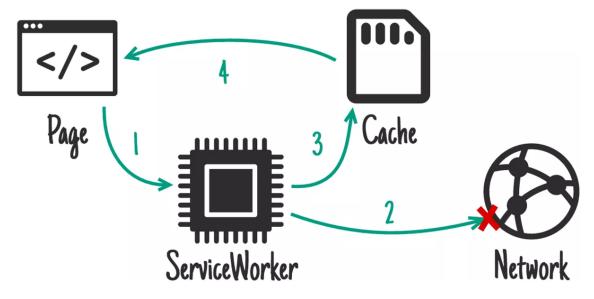


Abbildung 10: Network first (Theo 2021)

Cache-First ist das Pendant zu Network-First, setzt jedoch primär auf den Cache und nutzt das Netzwerk lediglich als Fallback. Dieser Ansatz eignet sich besonders bei einer schwankenden Netzwerkverbindung, ist aber nur dann zu empfehlen, wenn die Aktualität der Daten von unkritischer Natur ist. Das heisst, die Daten müssen entweder unerheblich oder beständig sein. Für dieses Projekt ist das Letztere der Fall. Einen Kompromiss dieser beiden Strategien zeigt die Abbildung 11 mit der Strategie Stale-While-Revalidate.

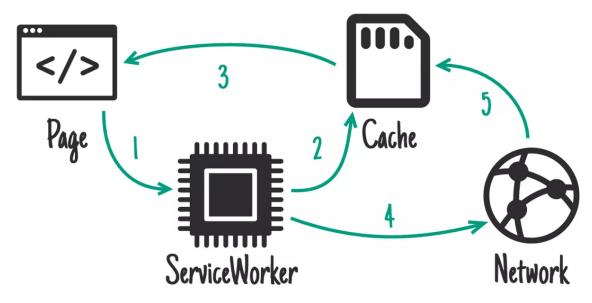


Abbildung 11: Stale-While-Revalidate (Theo 2021)

Dieser Ansatz zeigt einen kleinen Zyklus bei Abfragen, welche direkt über den Cache wieder auf die Seite gelangen – analog zu Cache-Only oder Cache-First. Direkt nach abgeschlossener Abfrage wird der Cache aber mit neuen Daten aus dem Netzwerk aktualisiert. Dadurch arbeiten sind die Daten stets so aktuell wie der Zeitpunkt der letzten Abfrage. Bei regelmässigen Abfragen kann daher von einer Aktualität der Daten gesprochen werden, insofern eine Netzwerkverbindung vorhanden ist. (Theo 2021)

Fazit

Welche dieser fünf Caching-Strategien für dieses Projekt am geeignetsten ist, kann zumindest teilweise anhand der Anforderungen (siehe 5.2) entschieden werden. Da die Lauffähigkeit offline (siehe 8.6.5) eine Anforderung ist, fällt die Strategie Network-Only bereits weg, weil dafür stets eine Netzwerkverbindung benötigt wird. Wie bereits bei Cache-Only angesprochen, bestehen für dieses Projekt kaum Vorteile, die nicht auch die übrigen Strategien zusätzlich zu ihrer Flexibilität bieten.

Die drei übrigen Caching-Strategien unterscheiden sich kaum in Bezug auf die Erfüllung der funktionalen Anforderungen (siehe 5.2.1), weshalb allesamt zur Auswahl stehen. Sie unterscheiden sich jedoch mit Blick auf Geschwindigkeit und Aktualität der Daten. Grundsätzlich gilt: Je aktueller die Daten desto langsamer ist die Abfrage. Die Methode State-While-Revalidating bildet dabei den Kompromiss, wo Daten einer ähnlichen Geschwindigkeit wie bei Cache-First abgefragt werden und trotzdem eine gewisse Aktualität auf Kosten der gesamten Performanz der Applikation aufweisen – immerhin muss die Abfrage immer doppelt ausgeführt werden. Es gilt also, die Relevanz von aktuellen Daten gegen die benötigten Ressourcen im Betrieb abzuwägen. Unter der Annahme, dass Letzteres bei einem Projekt dieser Grösse kaum von Belangen sein dürfte, sind regelmässige Aktualisierungen besonders beim Release einer solchen Applikation äusserst vorteilhaft, um fehlerhafte Daten unauffällig auszutauschen.

Mit Network-First ist die Aktualität der Daten weitgehend gesichert, wobei sich auch hier zwei Probleme ergeben: Einerseits kann die Aktualität bei einer unzureichender Netzwerkverbindung nicht garantiert werden, während der Nutzer über die Aktualität der Daten unaufgeklärt bleibt. Weil diese Problematik nicht kritisch für das Projekt ist und mit visuellen Meldungen dem Benutzer mitgeteilt werden kann, wird dieser Punkt weniger stark berücksichtigt. Kritischer ist die zweite Problematik: Die Verbindungsqualität eines Netzwerks ist nicht binär bewertbar, sondern als Spektrum zu verstehen. Deshalb besteht die Möglichkeit, dass schwache Netzwerkverbindungen vom Service-Worker als aktiv angesehen werden, die Abfragen jedoch eine überdurchschnittlich hohe Laufzeit aufweisen. (Ekwuno 2019)

Da die nicht-funktionale Anforderung bezüglich Performanz keine weiteren Spezifikationen aufweist, kann davon ausgegangen werden, dass dies unabhängig von der Verbindungsqualität zu verstehen ist. Aus diesem Grund wird diese Strategie nicht für das Projekt in Erwägung gezogen.

Als einzige Option bleibt daher State-While-Revalidating übrig. Aufgrund der Komplexität dieser Strategie und zeitlichen Gründen wird für dieses Projekt vorerst auf eine Implementation verzichtet. Stattdessen wird vorübergehend eine Abwandlung des Cache-First als Proof-of-Concept implementiert, bei welchem alle Ressourcen bereits am Anfang abgefragt und abgespeichert werden.

5.3.4 UI Framework

Konsistenz ist ein wichtiger Teil der visuellen Kommunikation, weshalb inzwischen viele Unternehmen auf einen Styleguide setzen. Obwohl die Erarbeitung eines Styleguides in der Regel ein langwieriger Prozess und somit für dieses Projekt out-of-scope ist, kann die Verwendung eines sogenannten User-Interface (kurz UI) Frameworks bereits ein strukturiertes Fundament legen, was als bewährte Vorgehensweise bei modernen Web-Applikationen gilt. (Lin 2020)

UI Frameworks stellen vorgefertigte Komponenten zur Verfügung, anhand welcher das Frontend einer Applikation gestaltet wird (Chernyakov 2021). Da eine Vielzahl der Komponenten wie Buttons, Formulare, Tabellen immer wieder Anwendung finden, bieten die meisten Frameworks eine gemeinsame Schnittmenge an – einen Standard. Je nach Framework unterscheiden sich die verfügbaren Komponenten und können abweichende Funktionen erfüllen. Deshalb ist es von zentraler Bedeutung, die benötigten Komponenten für das Projekt erst zu definieren, um sich früh auf ein UI Framework festzulegen.

Die Tabelle 1 zeigt eine Gegenüberstellung der zu manipulierenden Eigenschaften eines Charakters mit den benötigten UI Komponenten. Die Wahl der Komponenten ist so gewählt, dass die Benutzerfreundlichkeit maximiert wird. Besonders die Dropdowns mit Filterung und/oder Mehrfachauswahl stellen eine nutzerfreundliche Lösung zur Visualisierung und Manipulation des Charakters dar, weshalb diese mehrmals bei den Anforderungen erscheinen.

Tabelle 1: Komponenten und deren Anforderungen

| | Tabs | Option für Layout | Formular | Numerisches Feld | Textbox mit Validation | Dropdown mit Scroll | Dropdown mit Filter | Dropdown mit Mehrfachauswahl | Datagrid / Tabelle | File Upload |
|----------------------------|------|-------------------|----------|------------------|------------------------|---------------------|---------------------|------------------------------|--------------------|-------------|
| Charakterbogen | ✓ | ✓ | ✓ | - | - | - | - | - | - | ✓ |
| Attribute | - | - | - | ✓ | - | - | - | - | - | - |
| Personalien | - | - | - | ✓ | ✓ | ✓ | ✓ | - | - | - |
| Vor- und Nachteile | - | - | - | - | - | ✓ | ✓ | ✓ | - | - |
| (Kampf-)Sonderfertigkeiten | - | - | - | - | - | - | \checkmark | \checkmark | - | - |
| Talente | - | - | - | ✓ | - | - | - | - | - | - |
| Kampftechniken | - | - | - | ✓ | - | - | - | - | - | - |
| Zauber / Liturgien | - | - | - | ✓ | - | - | ✓ | ✓ | - | - |
| Inventar | - | - | - | - | - | - | - | - | ✓ | - |

Auf dieselbe Art und Weise zeigt die Tabelle 2 eine Auswahl von zehn UI Frameworks für Blazor, die eine kostenlose Verwendung erlauben (Malavasi 2020). Auf der Zeile markiert sind diejenigen Komponenten, welche das jeweilige UI Framework zur Verfügung stellt. Die Angaben basieren auf einer Analyse der bereitgestellten Komponenten des jeweiligen Anbieters. Bei den Frameworks ohne Beurteilung sind weder visuelle Beispiele noch eine Auflistung der verfügbaren Komponenten vorhanden.

Tabelle 2: UI Frameworks und deren Abdeckung von Anforderungen

| | Tabs | Option für Layout | Formular | Numerisches Feld | Textbox mit Validation | Dropdown mit Scroll | Dropdown mit Filter | Dropdown mit Mehrfachauswahl | Datagrid / Tabelle | File Upload |
|------------------------------|------|-------------------|----------|------------------|------------------------|---------------------|---------------------|------------------------------|--------------------|-------------|
| Radzen | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Blazorise | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | - | - | ✓ | - |
| Ant Design Blazer | ✓ | ✓ | ✓ | - | ✓ | - | - | - | ✓ | ✓ |
| BlazorStrap | ✓ | - | ✓ | - | ✓ | - | - | - | ✓ | - |
| PanoramicData.Blazor | - | ✓ | ✓ | - | - | - | - | - | ✓ | ✓ |
| MudBlazor | ✓ | ✓ | ✓ | ✓ | ✓ | - | - | - | ✓ | ✓ |
| Element Blazor (Auflistung) | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | - | - | ✓ | ✓ |
| Skclusive.Material.Component | ✓ | - | - | ✓ | ✓ | - | - | - | ✓ | - |
| MatBlazor | | | | | | | | | | |
| Blazored | | | | | | | | | | |

Die geringe Anzahl an verfügbaren Komponenten bei PanoramicData.Blazor ist darauf zurückzuführen, dass es primär ungewöhnliche Komponenten wie einen File Explorer anbietet und daher typischerweise in Kombination mit einem anderen Framework verwendet wird (Malavasi 2020).

Ausser Radzen bietet kein UI Framework ein Dropdown an, welches sowohl Filtersetzung als auch Mehrfachauswahlen erlaubt. Daraus lässt sich schliessen, dass diese Dropdowns über den Standard hinausgehen und daher eine innovative Lösung repräsentieren. Basierend auf dieser Erkenntnis und der Abdeckung an Standardkomponenten aus der Tabelle 2 eignet sich Radzen als UI Framework für dieses Projekt.

Aufgrund fehlender CSS-Klassen für eine vereinfachte Gestaltung von verschiedenen Mediengrössen wird zudem Bootstrap benutzt, was bereits standardmässig in Blazor integriert ist.

5.4 Roadblocks

Dieses Kapitel befasst sich mit den Roadblocks während des Projekts. Zur Erläuterung verschiedener Lösungen im Code wird in diesem Kapitel auf die einzelnen Komplikationen eingegangen, die über die täglichen Hürden eines Software-Entwicklers hinausgehen.

Weiterhin werden Abhängigkeiten des Projekts erläutert, um die Verwendung spezifischer NuGets anhand ihrer Einsatzmöglichkeiten zu begründen. Je nach Fall wird auch direkt auf den Code eingegangen. Die angegebenen Zeilennummern entsprechen denen des Endprodukts, insofern der Ausschnitt noch in dieser Form vorhanden ist. Ansonsten wird die Zeile angegeben, bei welcher sich der analoge Code befindet.

5.4.1 Debug & Cache

Je nach Browser werden Daten automatisch im Cache abgelegt, obwohl die Applikation neu über Visual Studio geöffnet wird. Dieses Problem trat während der gesamten Projektlaufzeit mehrmals auf. Besonders bei kleinen Anpassungen im HTML und/oder CSS wurden diese nicht immer aktualisiert, wodurch der Zeitaufwand für entsprechende Anpassungen anstieg.

Um dieses Problem zu umgehen, bieten Browser in der Regel eine Option, die es erlaubt, den Cache beim Beenden des Browsers zu leeren. Falls dies unerwünscht ist, kann dies auch manuell vorgenommen werden.

Dieser Workaround war aber nicht in der Lage, alle Caching Probleme zu beheben. Als der Service-Worker für die PWA eingesetzt wurde, entstanden weitere Probleme in Bezug auf Caching: Der in Visual Studio 2019 integrierte Web-Server Internet Information Service (kurz IIS) hat einen eigenen Cache, der nicht über den Browser bereinigt werden kann. Gelöst wurde dies durch die Abänderung des Ports innerhalb der Datei «launchSettings.json» im Projekt.

5.4.2 Mock für Testing

Einzelne Blazor-Komponenten benötigen Zugriff auf den wwwroot über den HttpClient. Für das Testing all jener Komponenten wird ein Mock des HttpClients benötigt, um die Kontrolle über die Anfragen zu erhalten und mit eigenen Inhalten zu beantworten. Weil der HttpClient als Singleton via Dependency Injection in die Komponenten gelangt, ist das Ersetzen des Services mit wenig Aufwand verbunden – auch bei Tests.

Hierfür wurde zusätzlich das NuGet «RichardSzalay.MockHttp» dem Projekt hinzugefügt, was vordefinierte Antworten auf Web-Requests erlaubt. Im Grunde genommen handelt es sich dabei also um einen Stub und nicht einen Mock (Ponte 2018). Zur einfacheren Handhabung wurde ausserdem im Code eine zusätzliche Extension-Class geschrieben. Dadurch wird die Klasse «MockHttpMessageHandler» mit neuen Methoden erweitert, die in den automatisierten Tests Verwendung finden.

5.4.3 Polymorphie mit Json

Eine weitere Problematik ist die Polymorphie bezüglich Imports und Exports des Inventars als Bestandteil des Charakterbogens. Im Datenmodell wird das Inventar als Liste vom Typ «Item» deklariert, was generell kein Hindernis darstellt. Weil Item aber als Basisklasse für abgeleitete Klassen wie «Weapon» fungiert, führt das zu Unterschieden bei der Anzahl an Elementen nach einem Export. Der Code 2 zeigt diesen Sachverhalt anhand eines exportierten Charakterbogens auf: Beide Objekte in dieser Liste sind grundsätzlich Items aber nur das zweite Objekt ist auch eine Waffe, was zusätzliche Elemente zur Folge hat.

Code 2: Liste von Items als Json

```
"Items": [
01
02
          {
                 "$type": "DSA.Inventory.Item, DSA",
03
                 "Name": "Item1",
04
                 "Price": 1,
05
06
                 "Weight": 1,
07
                 "StructurePoints": 10
98
          },
09
                 "$type": "DSA.Inventory.Weapon, DSA",
10
                11
12
                       "Number": 1,
13
14
                       "Sides": 6
15
                 "ModTP": 1,
16
                 "ModAT": 1,
17
18
                 "ModPA": -1,
19
                 "Reach": 1,
20
                 "CombatTechnique": "Dolche",
21
                 "BonusDamageThreshold": 14,
                 "WeaponAdvantage": null,
22
23
                 "WeaponDisadvantage": null,
                 "Name": "Weapon1",
24
25
                 "Price": 11,
26
                 "Weight": 11,
27
                 "StructurePoints": 11
28
          }
29 }
```

Das NuGet «Newtonsoft» ist für das Element «\$type» verantwortlich. Ohne dessen Verwendung mit der Konfiguration von «TypeNameHandling.Objects» würden die Zeilen 03 und 10 nicht existieren. Folglich ist nicht klar identifizierbar, um welche Typen es sich bei den beiden Objekten handelt. Daher ist auch kein Import ohne entsprechende Definition möglich, weil nicht alle vorhandenen Objekte als Objekte vom Typ «Item» interpretiert werden können.

Im Kapitel Validierung der Voraussetzungen (siehe 5.4.5) wird diese Problematik erneut von einer anderen Seite beleuchtet und zusätzlich erläutert, warum die angewandte Lösung nicht zufriedenstellend ist.

5.4.4 Multi-Level Chained Bind

Damit eine Validierung des Charakters bei jeder Änderung ausgelöst wird, muss das Objekt über jegliche Änderungen informiert werden. Grundsätzlich übernimmt Blazor diese Aufgabe, wenn Objekte mit @bind als Parameter an Blazor-Komponenten übergeben wird. Der Code 3 zeigt die Anwendung am Beispiel des «AttributesComponent», verwendet im «CharacterComponent» (früher Stand).

Code 3: Two-Way-Binding in HTML - Parent

Aufgrund dieses Bindings verlangt Blazor eine entsprechende Ergänzung im Code der Child-Komponente. Die Anpassungen im Projekt sind im Code 4 aufgezeigt. Vorzufinden sind zwei Properties. Die Zeile 50 beschreibt das Property für den Zugriff des Parameters, während Zeile 51 einen Event-Handler beschreibt. Im Allgemeinen wird dieser verwendet, um die Parent-Komponente über Änderungen zu informieren.

Code 4: Two-Way-Binding in HTML - Child

Blazor löst den EventCallback nach einer Änderung automatisch aus. Bei einem Multi-Level Binding ist jedoch Vorsicht geboten. Multi-Level Binding beschreibt ein mehrstufiges Binding: Analog zum Code 3 auf Zeile 12 wird von der Grandparent-Komponente ein Binding zur Parent-Komponente erstellt, wodurch sich das Binding über mehr als zwei Stufen erstreckt.

Die Abbildung 12 stellt das Verhältnis zwischen Grandparent-, Parent- und Child-Komponente visuell dar. Für die Validierung der Voraussetzungen (siehe 5.4.5) ist es erforderlich, dass die EventCallbacks verkettet (chained) sind, indem «AttributesChanged» zwingend «CharacterChanged» nach sich zieht.

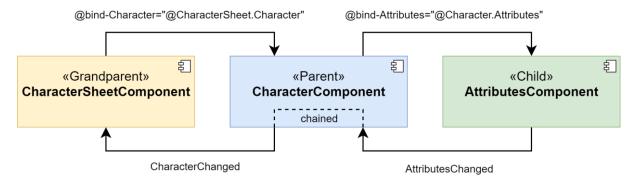


Abbildung 12: Multi-Level Chained Binding

Der bisherige Code erfüllt diese Anforderung noch nicht. Stattdessen wird «AttributesChanged» ausgelöst und lediglich «CharacterComponent» darüber informiert, ohne dieses Event weiterzureichen. Das Problem hierfür liegt im Binding: Die Parent-Komponente wird über die Änderung eines Parameters (hier Attributes) benachrichtigt, realisiert jedoch nicht, dass der Parent (hier Character) dadurch indirekt verändert wurde.

Um diese Kette von Ereignissen über beliebig viele Komponenten aufrecht zu erhalten, werden in diesem Projekt ausschliesslich Wrapper für solche Parameter verwendet. Der Code 5 zeigt die angepasste Version von Code 3.

Code 5: Two-Way-Binding in HTML - Parent (angepasst)

Das Binding wird nun mit einem neuen Property «AttributesWrapper» vorgenommen. Dieses wird im Code 6 gezeigt, wie es im Endprodukt im Einsatz ist. Anhand des Codes ist zu erkennen, dass der Wrapper nur eine Funktion aufweist, nämlich den EventCallback «CharacterChanged» aufzurufen, sobald Änderungen in der Child-Komponente auftreten.

Code 6: Property AttributesWrapper

```
public Ienumerable<DSA.Character.Attribute> AttributesWrapper
50
51
    {
52
          get => Character.Attributes;
53
          set
54
          {
55
                 Character.Attributes = value;
56
                 CharacterChanged.InvokeAsync(Character);
57
          }
58 }
```

Analog dazu werden auch alle anderen Parameter, sofern sie mehrstufig sind, mit einem Wrapper realisiert. Seither sind keine weiteren Probleme bei der Kaskadierung von Änderungen bekannt.

5.4.5 Validierung der Voraussetzungen

Ausgehend von den funktionalen Anforderungen (siehe 5.2.1) wird eine Validierung der Voraussetzungen benötigt, um die Integrität des Charakters zu gewährleisten. Dabei kann zwischen zwei verschiedenen Arten von Regeln unterschieden werden, die eine Validierung benötigen. Zum einen sind das allgemeine Regeln, die jeder Charakter einhalten muss: Beispielsweise dürfen die Kosten aller Fähigkeiten nicht das Total der Erfahrungspunkte überschreiten. Weil diese Voraussetzungen statisch sind, werden sie unabhängig vom Charakter überprüft.

Zum anderen können lernbare Fähigkeiten wie Vor- und Nachteile und alle Arten von Sonderfertigkeiten eigene Voraussetzungen aufweisen, die validiert werden müssen: Die allgemeine Sonderfertigkeit «Abrollen» benötigt beispielsweise eine Körperbeherrschung von 4 und der Charakter darf den Nachteil «Fettleibig» nicht aufweisen. Da die Voraussetzungen all dieser Fähigkeiten erst durch deren Spezifikation klar sind und beliebig viele sein können, werden die Voraussetzungen im Code als Liste innerhalb jeder Spezifikation geführt. Hierbei werden nur diejenigen validiert, die für den Charakter selektiert sind und dadurch nicht allgemeingültig beziehungsweise dynamisch sind.

Die Abbildung 13 zeigt die statischen und dynamischen Anforderungen zusammen mit deren Quelle auf. Zu erkennen ist, dass sowohl Vor- und Nachteile wie auch Sonderfertigkeiten eine Vielzahl von Eigenschaften als Voraussetzung haben können. Es sind sogar Beziehungen auf sich selbst möglich: Sonderfertigkeiten setzen andere Sonderfertigkeiten voraus. Damit können auch zirkuläre Abhängigkeiten entstehen, welche bei einer manuellen Erweiterbarkeit von Eigenschaften zu Problemen führen kann.

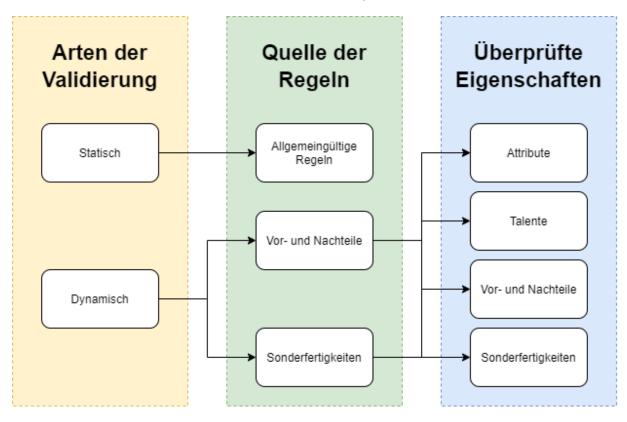


Abbildung 13: Aufteilung der Anforderungen

Auf die überprüften Eigenschaften der allgemeingültigen Regeln wird aufgrund der Komplexität nicht im Detail eingegangen, zumal der Code dafür ohnehin hartkodiert ist und selten angepasst werden muss. Im Gegensatz dazu müssen die Voraussetzungen für die Spezifikationen (beziehungsweise Vor-/Nachteile und Sonderfertigkeiten) zusammen mit den restlichen Daten abgelegt werden. Daher stammt die Bezeichnung dynamisch, weil die Voraussetzungen abhängig von den hinzugefügten Spezifikationen auf dem Charakter sind und je nach Regeln unterschiedliche Validierungen benötigen.

Hierfür bietet sich eine Liste mit Objekten vom Typ «Validator» auf der Spezifikation an, wobei die einzelnen Validators (Attribut, Talent, Vor-/Nachteil, Sonderfertigkeit) als eigene Klassen programmiert sind und ein Interface implementieren oder von einer gemeinsamen Klasse erben.

Dieser Ansatz führte jedoch zu Problemen mit der Persistenz: Json ermöglicht typischerweise keine Polymorphie, obwohl Packages wie «Newtonsoft» (siehe 5.4.3 Polymorphie mit Json) dies zur Verfügung stellen. Diese Eigenschaft wurde aber beim NuGet «System.Text.Json» (Microsoft) bewusst weggelassen, um Sicherheitsrisiken bei Web-Applikationen vorzubeugen (GrabYourPitchforks 2019). Beim Umgang mit nicht vertrauenswürdigen Daten, was bei jeder öffentlichen API der Fall ist, wird von der Interpretation von Typinformationen abgeraten:

«Serializers are security sensitive APIs and should not be used with untrusted data. This is not a problem specific to Java serialization, a specific .NET formatter or any specific formats such as JSON, XML or Binary. All serializers need to reconstruct objects and will normally invoke methods that attackers will try to abuse to initiate gadget chains leading to arbitrary code execution. » (Muñoz and Mirosh 2017)

Aus diesem Grund wird hier auf eine Serialisierung mit Typinformationen verzichtet und für jede Art von Voraussetzung eine eigene Liste geführt. Dieser Ansatz verursacht einen Overhead bei der Serialisierung und Validierung, was zu weniger Performanz führt. Zudem erhöht sich die Grösse des serialisierten Objekts, wobei dies kaum einen bemerkbaren Einfluss auf die Gesamtgrösse hat.

Als Konsequenz muss auch die Serialisierung des Charakters beim Import/Export erneut evaluiert werden, ob das Risiko durch Typinformationen eingegangen wird. Benötigt wird diese Typinformation bei den Items im Inventar (siehe 5.4.3 Polymorphie mit Json).

Service zur Validierung

Für die Abarbeitung aller Voraussetzungen wird der Applikation ein Service vom Typ «ValidationService» als Singleton hinzugefügt. Dadurch ist die Validierung sowie dessen Resultate auf der Web-Applikation auf jeder Komponente via Dependency Injection abrufbar. Gleichzeitig erhöht sich die Wiederverwendbarkeit, Wartbarkeit und Testbarkeit durch eine losere Kopplung (Trivedi 2020).

Derzeit erlaubt der Service die Validierung eines Charakters und eine Abfrage der Resultate, die als Error ausgegeben werden. Diese Implementation dient zudem als Schnittstelle zwischen dem Charakter, der bei jeder Änderung eine Validierung von sich selbst auslöst, und einer Anzeige der Resultate. Nach Bedarf kann der Service mit weiteren Abfragen von spezifischen oder auch allen Resultaten erweitert werden.

6 Evaluation und Validation

Dieses Kapitel dient als Standortbestimmung für das Produkt am Ende des fünften Sprints. Der Fokus liegt dabei auf der Beurteilung der Anforderungen (siehe 5.2) und inwiefern diese erfüllt sind. Zusätzlich wird eine Neubewertung der Anforderungen, basierend auf externem Feedback, vorgenommen.

6.1 Alpha-Testing

Um die nur schwer überprüfbaren, nicht-funktionalen Anforderungen (siehe 5.2.2) zu validieren, wird auf ein Alpha-Testing zurückgegriffen. Hierfür wird die Applikation als standalone Blazor WebAssembly auf einem privaten Server gehostet und so den Teilnehmenden zur Verfügung gestellt. Anschliessend werden alle Teilnehmenden gebeten, den dazugehörigen Fragebogen (siehe 8.7) wahrheitsgetreu auszufüllen.

Der Begriff Alpha-Testing wird in diesem Kontext als eine frühzeitige Überprüfung des Produkts durch Dritte definiert, mit dem Ziel, Erkenntnisse zum Stand des Produkts zu erhalten: Nebst ungewolltem Verhalten der Applikation werden gleichzeitig die Priorität der einzelnen Anforderungen sowie die Anforderungsliste als Ganzes erneut bewertet. In diesem Fall übernimmt die persönliche Spielgruppe, statt wie üblicherweise interne Mitarbeiter, diese Verantwortung. (Freiler 2019)

Der Fragebogen zielt darauf ab, wertvolles Feedback von potenziellen Kunden zu erlangen. Als Angehörige dieses Zielpublikums hat die eigene Spielgruppe, bestehend aus fünf weiteren Individuen, einen ersten Blick auf das Produkt geworfen und den dazugehörigen Fragebogen (siehe 8.7) wahrheitsgetreu ausgefüllt. Die aggregierten Resultate (siehe 8.7.1) befinden sich direkt im Anschluss.

6.1.1 Überprüfung der nicht-funktionalen Anforderungen

Anhand der ersten vier Fragen werden die nicht-funktionalen Anforderungen durch die Teilnehmenden mit einem Wert zwischen 1 und 4 bewertet. Die Tabelle 3 zeigt das Resultat: Die Akzeptanz ist in Bezug auf die Performanz und Verständlichkeit überdurchschnittlich hoch. Im Kontrast dazu besteht noch Potenzial in der Benutzerführung, da die Applikation von drei Befragten als nicht selbsterklärend empfunden wurde.

Tabelle 3: Resultat der Fragen zu den nicht-funktionalen Anforderungen

| | Nein | Eher nein | Eher ja | Ja |
|---|------|-----------|---------|----|
| Ist die Applikation für dich übersichtlich gestaltet? | 0 | 0 | 4 | 1 |
| Empfindest du die Aufteilung der Inhalte als sinnvoll? | 0 | 0 | 2 | 3 |
| Findest du die Applikation selbsterklärend? | 3 | 0 | 2 | 0 |
| Hat die Applikation deine Anpassungen ohne drastische Verzögerung übernommen? | 0 | 0 | 1 | 4 |
| Würdest du dieses Produkt nach Release verwenden? | 1 | 1 | 1 | 2 |

Undeutlich zu interpretieren sind die Bewertungen zur Frage, ob die Teilnehmenden das Produkt nach Release verwenden würden. Entweder wurde die Frage nicht von allen auf dieselbe Art und Weise verstanden, oder die Ansichten gehen hierfür weit auseinander. Es lässt sich vermuten, dass die Absenz von Funktionalitäten die Bewertung dieser Frage negativ beeinträchtigt hat, obwohl die Vollständigkeit in der Frage durch «nach Release» impliziert wird. Wahrscheinlich wurde dieser Sachverhalt aber nicht genügend deutlich vermittelt. Aufgrund der unklaren Datenlage wird deshalb für diese Frage auf eine tiefgreifende Interpretation verzichtet.

6.1.2 Neubewertung der Anforderungen

Im zweiten Teil des Fragebogens (siehe 8.7) wird den Teilnehmenden ermöglicht, noch nicht implementierte Anforderungen nach ihrer Wichtigkeit zu bewerten. Hierfür wird, wie bei den vorhergehenden Fragen, bewusst eine Skala von 1 bis 4 gewählt. Damit wird gewährleistet, dass jede Antwort eine Tendenz aufweist, aber trotzdem Abstufungen vorhanden sind. Die Tabelle 4 zeigt wiederum das aggregierte Resultat.

Tabelle 4: Resultat der Fragen zu den funktionalen Anforderungen

| | Irrelevant | Unwichtig | Wichtig | Essenziell |
|--|------------|-----------|---------|------------|
| Manuelle Erfassung von eigenen Inhalten | 0 | 0 | 2 | 3 |
| Standardisierter Json-Export als Schnittstelle | 0 | 0 | 2 | 3 |
| Standardisierter Json-Import als Schnittstelle | 0 | 3 | 0 | 2 |
| Spielunterstützung mit Schaden, Status, etc. | 2 | 3 | 0 | 0 |
| Möglichkeit direkt in der Applikation zu würfeln | 2 | 3 | 0 | 0 |

Zusammen mit der darauffolgenden Frage, was sonst noch in der Applikation vermisst wird, schaffen die Bewertungen die Grundlage für eine Neubewertung der Anforderungen hinsichtlich ihrer Priorität. Unter Anbetracht der Resultate zeigen sich klare Tendenzen bei den bewerteten Anforderungen ab: Sowohl die manuelle Erfassung von eigenen Inhalten (siehe 8.6.8) wie auch der standardisierte Json-Export (siehe 8.6.9) werden von allen Teilnehmenden als wichtig oder gar essenziell angesehen. Dies bestätigt die Aktualität der Anforderungsliste, die zu Beginn dieser Arbeit erarbeitet wurde.

Anhand der Spielunterstützungsfunktionalitäten und der Möglichkeit, direkt in der Applikation zu würfeln, bestimmt sich der Scope der Applikation. Weil beide dieser Anforderungen ausschliesslich mit unwichtig oder irrelevant bewertet wurden, bleibt die primäre Funktion der Applikation zur Charaktererstellung und -manipulation bestehen.

Nebst fehlenden Funktionalitäten im Inventar oder fehlender Inhalte im Allgemeinen wurden noch zwei weitere Anforderungen erwähnt. Auf der einen Seite fehlen noch spieltechnisch relevante Erklärungen zu den Inhalten. Diese Angaben sind zwar in den Daten vorhanden, werden aber (noch) nicht ausgegeben. Zum anderen ist eine zusammengefasste Druckversion erwünscht. Der Grund hierfür besteht vermutlich in der Präferenz, traditionell mit Stift und Papier anstatt eines technischen Hilfsmittels zu spielen.

6.1.3 Überprüfung der Anforderungen

Zur Überprüfung der funktionalen Anforderungen werden die Antworten zur Frage bezüglich Crashes, Anzeigefehler oder anderweitigen Problemen mit den bestehenden Anforderungen abgeglichen, um diese gegebenenfalls zu falsifizieren.

Aus den Antworten zum Fragebogen (siehe 8.7.1) ist zu entnehmen, dass der Charakter unerwarteterweise verworfen wird, sobald auf «Home» navigiert wird. Dies wird als kritischer Konflikt mit der Anforderung der Persistenz des Charakters (siehe 8.6.3) angesehen und sollte schnellstmöglich behoben werden.

Eine Unschönheit, welche jedoch nicht als kritisch angesehen wird, ist die doppelte Fehlermeldung, wenn zu viele Punkte vergeben wurde. Nach einer oberflächlichen Fehleranalyse konnte dieses Verhalten auch nicht reproduziert werden. Ein weiterer Fehler von rein visueller Natur ist die Möglichkeit, ungültige Werte bei numerischen Feldern einzugeben. Hierfür muss direkt hintereinander ein ungültiger Wert manuell eingegeben werden, damit das Feld den letzten Wert anzeigt – im Modell wird jedoch keine Anpassung vorgenommen. Diese Inkonsistenz scheint vom verwendeten UI Framework (siehe 5.3.4) zu stammen.

6.2 Validation durch Testkonzept

Zusätzlich zum Alpha-Testing wird zur Validation der funktionalen Anforderungen (siehe 5.2.1) das jeweils letzte Resultat der manuellen und automatisierten Tests herangezogen. Sowohl die manuellen als auch automatisierten Tests wurden allesamt erfolgreich ausgeführt. Daraus lässt sich schliessen, dass die absolute Wahrscheinlichkeit für einen offensichtlichen Fehler klein ausfällt. Durch die Kombination aus manuellen und automatisierten Tests wird das Datenmodell extensiv getestet, allerdings ist dadurch noch keine vollumfängliche Fehlerfreiheit garantiert.

Aufgrund der DoD (siehe 4.2 Projekt- und Vorgehensmodell) muss für jedem Arbeitspaket bei Abschluss die CI erfolgreich durchlaufen. Aus diesem Grund wird auf die Resultate der automatisierten Tests nicht weiter eingegangen. Die Tabelle 5 zeigt eine Übersicht der realisierten Anforderungen und inwiefern diese durch automatisierte und/oder manuelle Tests verifiziert werden. Sämtliche Funktionalitäten, beruhend auf funktionalen Anforderungen, werden also mindestens durch manuelle Tests überprüft.

Tabelle 5: Übersicht der Anforderungen und deren Verifizierung

| | automatisiert | manuell |
|--|---------------|--------------|
| Charaktererstellung und -manipulation | √ | √ |
| Persistenz des Charakters | - | ✓ |
| Keine Installation notwendig | - | ✓ |
| Lauffähig offline | - | \checkmark |
| Berechnungen sind korrekt und komplett | ✓ | ✓ |
| Validierung der Abhängigkeiten | ✓ | \checkmark |

6.2.1 Bewertung durch Google Lighthouse

Um die beiden Anforderungen Lauffähigkeit offline (siehe 8.6.5) und keine Installation notwendig (siehe 8.6.4) zu überprüfen, gibt es unter Testdrehbücher eine Anleitung zur Überprüfung dieser Anforderungen. Anstatt diese Analyse selbst durchzuführen, wird jedoch Google Lighthouse verwendet. Dieses Werkzeug überprüft, ob die inspizierte Seite alle Qualifikationen für eine PWA erfüllt. Das heisst, die Applikation wird auf die Performanz, Zugänglichkeit, bewährte Praktiken, Suchmaschinenoptimierung (kurz SEO) und die Installierbarkeit überprüft (Pospelova 2019). Die Abbildung 14 zeigt die vier Ergebnisse der zuletzt durchgeführten Prüfung. Unter Verbesserungsvorschläge (siehe 8.8) befinden sich Auszüge aus dem Report der Prüfung zur Verbesserung der Applikation.



Abbildung 14: Ergebnisse von Google Lighthouse

Auffällig ist das unterdurchschnittliche Ergebnis zur Performanz. Dieses hängt stark mit der angewandten Technologie zusammen, denn die Mehrheit der Verbesserungsvorschläge für die Performanz (siehe 8.8.1) bezieht sich direkt auf Blazor WebAssembly.

Diese Beobachtung wird dadurch bestätigt, dass Blazor WebAssembly im Vergleich zu Angular durchaus schlechter abschneidet, solange keine Komprimierung der Inhalte durch den Server vorgenommen wird. Durch die Komprimierung hat sich die anfängliche Ladezeit bei Blazor WebAssembly von rund 19,5 Sekunden auf 1,6 Sekunden verkürzt, während bei Angular kein Unterschied zu erkennen ist. Die Tabelle 6 zeigt diese Ergebnisse mit und ohne Komprimierung. Die Werte basieren auf der Standard-Applikation der jeweiligen Technologie. (Bourgarel 2020)

Tabelle 6: Vergleich bezüglich Performanz zwischen Blazor und Angular (Bourgarel 2020)

| | Blazor WebAssembly | Angular |
|-------------------------------|--------------------|---------|
| Performanz ohne Komprimierung | 65 | 98 |
| Performanz mit Komprimierung | 100 | 98 |

Des Weiteren weisen auch Zugänglichkeit und SEO noch Verbesserungspotenzial auf. Bei beiden scheint sich ein Grossteil der Vorschläge auf die Präsentation (HTML und CSS) zu konzentrieren. Eine Auflistung der Verbesserungsvorschläge für die Zugänglichkeit (siehe 8.8.2) respektive SEO (siehe 8.8.4) befindet sich im Anhang. Im darauffolgenden Kapitel befinden sich die Verbesserungsvorschläge für eine optimierte PWA (siehe 8.8.5), wobei auch hier ein Element im HTML vermisst wird. Nebenbei fehlt im Manifest ein sogenanntes maskierbares Symbol zur Darstellung von Transparenz in Icons (Oakes and Steiner 2021).

6.3 Konkurrenzvergleich

Zu einer Standortbestimmung gehört auch eine Analyse der Hauptkonkurrenten, inklusive deren Stärken und Schwächen. Daraus können sich Chancen und Risiken für das Produkt ergeben, die beim Erfolg des Projekts einen entscheidenden Faktor spielen können. (Athuraliya 2021)

Die Tabelle 7 zeigt hierzu eine Gegenüberstellung der zuvor identifizierten Hauptkonkurrenten (siehe 2.1 Competitive Landscape) und der technischen Anforderungen, inklusive der Wünsche aus dem Fragebogen (siehe 8.7). Ein Häkchen in Klammern bedeutet, dass die Anforderung nur ansatzweise erfüllt sind.

Tabelle 7: Konkurrenzvergleich anhand der funktionalen Anforderungen

| | Charaktererstellung und -manipulation | Persistenz des Charakters | Keine Installation notwendig | Lauffähig offline | Berechnungen sind korrekt und komplett | Validierung der Abhängigkeiten | Manuelle Erweiterbarkeit | Standardisierter Json-Export | Standardisierter Json-Import | Ausgabe spieltechnischer Erklärungen | Übersichtliche Druckversion |
|----------------------------|---------------------------------------|---------------------------|------------------------------|-------------------|--|--------------------------------|--------------------------|------------------------------|------------------------------|--------------------------------------|-----------------------------|
| Endprodukt dieser Arbeit | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | (√) | - | - | - | - |
| Deluxe Heldendokument | ✓ | ✓ | ✓ | ✓ | ✓ | - | (√) | - | - | - | \checkmark |
| Foundry VTT | ✓ | ✓ | ✓ | - | ✓ | - | (√) | - | ✓ | (√) | - |
| Optolith Character Creator | ✓ | ✓ | - | ✓ | ✓ | ✓ | - | ✓ | ✓ | ✓ | ✓ |
| The Dark Aid | ✓ | ✓ | - | ✓ | ✓ | ✓ | - | - | - | ✓ | - |

Basierend auf der Gegenüberstellung können fehlende Funktionalitäten der einzelnen Konkurrenzprodukte identifiziert werden. Zusammen mit den Ergebnissen aus dem Alpha-Testing (siehe 6.1) bezüglich der Wichtigkeit von Anforderungen bietet sich die Möglichkeit, Chancen und Risiken daraus abzuleiten.

Die manuelle Erweiterbarkeit wird durch die Teilnehmenden als überdurchschnittlich wichtig angesehen, trotzdem implementiert keines der Produkte diese Funktionalität vollumfänglich. Hier besteht also eine Marktnische, die bei der richtigen Umsetzung zum Erfolg führen kann. (Montag 2021)

In diesem Fall überwiegen aber die Risiken: Der Standardisierte Json-Export erhielt die gleiche Bewertung wie die manuelle Erweiterbarkeit und wurde bereits vom Optolith Character Creator integriert. Falls dieser Export nicht vom Produkt bereitgestellt wird, läuft man Gefahr, potenzielle Kunden zu verlieren. Dasselbe gilt für die Ausgabe spieltechnischer Erklärungen und der Möglichkeit einer übersichtlichen Druckversion mit dem Unterschied, dass die Wichtigkeit dieser beiden Anforderungen nicht genau bestimmt wurde.

7 Ausblick

In diesem Kapitel wird auf die anschliessenden Schritte in der Produktentwicklung und einen persönlichen Rückblick der gesammelten Erfahrungen eingegangen.

7.1 Weiteres Vorgehen

Das Endprodukt ist aus unterschiedlichen Gründen noch nicht bereit für eine Markteinführung. Zum einen ist die Distribution in Zusammenarbeit mit Ulisses noch offen. Glücklicherweise kann das Produkt sowohl als lokale Applikation via Download als auch als PWA angeboten werden, wodurch eine Festlegung zu diesem Zeitpunkt nicht erforderlich ist. Weiterhin müssen noch Kernfunktionen zur Charaktererstellung und -manipulation ergänzt (Auswahl in Dropdowns, Inventar) beziehungsweise erst hinzugefügt werden.

Das Alpha-Testing (siehe 6.1) hat einen Einblick in den Stand des Produkts gewährt. Basierend auf den Rückmeldungen werden in einem weiteren Schritt die Prioritäten der Anforderungen neu evaluiert und bei Bedarf neue Arbeitspakete dazu im Backlog erfasst. Ein Beispiel für eine solche Anforderung könnte die Überarbeitung der Benutzeroberfläche sein, um eine engere Kundenbindung zu erzeugen. Ebenso werden die Verbesserungsvorschläge aus der Prüfung mit Google Lighthouse (siehe 8.8) bewertet und allenfalls in die zukünftigen Sprints eingeplant, indem sie priorisiert im Backlog erfasst werden. Im gleichen Zug werden die identifizierten Probleme (Sicherheitsrisiko bei typisierten Objekten in Json) im Backlog aufgenommen.

Sobald das Produkt die Kernfunktionalitäten aufweist, dient ein Beta-Testing als nächsten Meilenstein. Wie beim Alpha-Testing lohnt sich hierbei ebenfalls die Zielsetzung der Fragen im Voraus zu bestimmen, damit die entsprechenden Bereiche der Applikation zur Verfügung gestellt werden. Gegebenenfalls lohnt sich dafür ein geschlossenes Beta-Testing mit Teilnehmenden aus einer spezifischen Demographie oder erwünschten Fähigkeiten (Babich 2019). Im Gegensatz zum Alpha-Testing ist die Auswahl der Teilnehmenden von essenzieller Bedeutung, um die Objektivität zu gewährleisten. Hierfür eignen sich fremde Personen aus dem Zielpublikum (Freiler 2019).

Die erhaltenen Rückmeldungen werden analog zum Alpha-Testing analysiert und interpretiert, um daraus einen Mehrwert zu gewinnen. Kritische Belange werden adressiert, bevor das Produkt zur Markeinführung übergeht – dies stellt einen weiteren Meilenstein in der Projektplanung dar. Bei einer hohen Anzahl an kritischen Problemen ist eine Repetition des Beta-Testings denkbar, um das Risiko bei der Markteinführung zu minimieren.

Spätestens bei der Markteinführung muss die Frage der Distribution geklärt sein. Der Beschluss beeinflusst das Vorgehen insofern, dass bei einem Hosting entsprechende Infrastruktur benötigt wird. Zusätzlich muss die Frage bezüglich der Software-Architektur (siehe 5.3.1) noch geklärt werden: Braucht es einen Server oder reicht standalone Blazor? Aus Gründen der Qualitätssicherung ist ein Entscheid vor dem Beta-Testing sinnvoll, damit die Architektur als Bestandteil der Applikation überprüft wird.

7.2 Reflexion

Im Verlauf dieser Arbeit wird bereits ein beträchtlicher Teil des Gesamtumfangs für die Veröffentlichung eines modernen Charaktereditors für DSA abgedeckt. Dazu gehören mitunter fundamentale Konzepte bezüglich der Architektur und technische Lösungsansätze für die Implementation von Anforderungen.

Bei den ungelösten Problemen wird ausserdem in Aussicht gestellt, inwiefern eine zeitnahe Abklärung sinnvoll ist und welche Informationen dafür noch benötigt werden. Zusammen mit den Rückmeldungen aus dem Alpha-Testing bietet daher die Arbeit ein stabiles Fundament für die Weiterentwicklung des Produkts.

Eine bessere Bewertung durch Google Lighthouse in Bezug auf Performanz (siehe 8.8.1) hätte ich sehr begrüsst. Zugleich freut mich aber die positive Bewertung in anderen Kategorien, in denen die angewandte Technologie weniger Einfluss hat. Letztendlich unterstützt Google Lighthouse die technische Optimierung von Web-Applikationen auf einfachste Art und Weise, die mir vor diesem Projekt nicht bekannt war.

7.2.1 Projektmanagement

Vor Beginn des Projekts waren meine Erwartungen an das Endprodukt höher als der Stand beim Abschluss. Der Zeitaufwand wurde also gesamthaft unterschätzt, während der Aufwand der einzelnen Sprints – mit Ausnahme des zweiten Sprints – kaum unterschätzt wurde. Diese Beobachtung unterstützt die Auffassung, dass eine Schätzung zum zeitlichen Aufwand eines Arbeitspaketes schwieriger wird, umso umfangreicher das zu schätzende Arbeitspaket ist (Radigan 2021). Obwohl die Sprints nicht immer komplett abgearbeitet wurden, konnten die einzelnen Arbeitspakete oftmals im geplanten Umfang realisiert werden. Das Problem waren nicht überfüllte Sprints, sondern die persönlichen Anforderungen an den Code (Refactoring) und Anpassungen im UI, die im Planning nicht als Teil des Sprints eingeplant wurden.

Während der Initialisierungsphase hätte ich noch mehr Zeit in die Recherche zu Blazor WebAssembly und PWA investieren können. Allenfalls hätte dadurch der erste Entwurf der Software-Architektur anders ausgesehen beziehungsweise die Frage bezüglich standalone oder hosted Blazor wäre bereits vorab aufgekommen und bereits innerhalb der Arbeit geklärt worden. Auch wenn es unbefriedigend ist, dass diese Frage offensteht, hatte sie in der Praxis kaum Einfluss auf die Realisierung des Projekts.

Schlussendlich hat sich ein hybrides Vorgehen wie SoDa für mich insofern gelohnt, dass ich mit einer realistischen Zeitplanung stets auf den nächsten Meilenstein hinarbeiten konnte. Diese Ausgangssituation unterstützt nicht nur meine persönliche Motivation, sondern hilft bei der Einhaltung von Terminen. Zudem hat sich der Planungsaufwand auf ein Minimum reduziert: Selbst bei der Initialisierungsphase hatte ich noch die Gelegenheit, an der Dokumentation zu arbeiten.

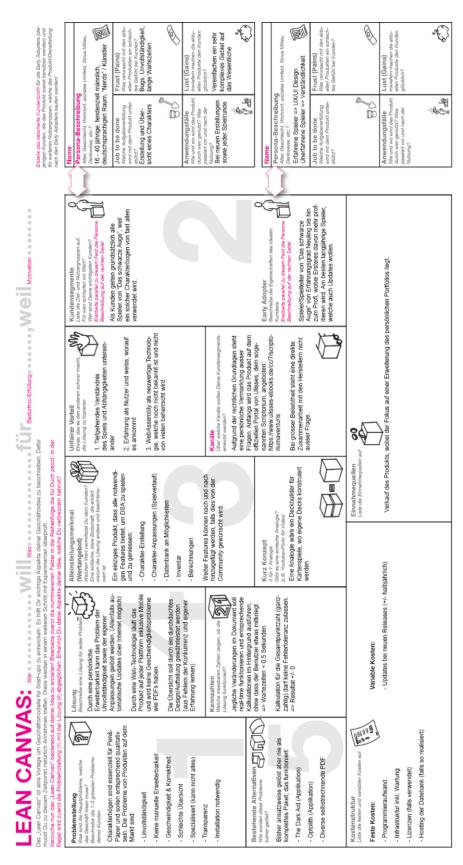
7.2.2 Fragebogen

Obwohl der Fragebogen für das Alpha-Testing einen höheren Mehrwert als erwartet generiert hat, war dieser nicht optimal ausformuliert: Die Fragestellungen waren nicht immer deutlich und entsprechend hat auch die Datenqualität darunter gelitten – als Resultat wurde die Bewertung einer Frage gänzlich ignoriert. Da ich die Teilnehmenden persönlich kenne, konnten einzelne Unklarheiten teilweise direkt beim Ausfüllen behoben werden. Deshalb kann aber noch nicht davon ausgegangen werden, dass bei allen Teilnehmenden die Frage zwingend gleich verstanden wurde.

Für zukünftige Projekte oder ein Beta-Testing werde ich mehr Zeit für die Ausarbeitung eines Fragebogens investieren, um eine möglichst hochwertige Datenqualität zu garantieren. Schlechte Daten führen zu schlechten Entscheidungen, was wiederum die Verschwendung von wertvollen Ressourcen zur Folge hat. (Shafei 2020)

8 Anhänge

8.1 Lean Canvas



8.2 Aufgabenstellung

Lucerne University of Applied Sciences and Arts

HOCHSCHULE LUZERN

Definitive Aufgabenstellung: Wirtschaftsprojekt / Bachelorarbeit

Informatik

1. Regulärer Starttermin: 20.09.2021 Individuelle Termine dürfen direkt angepasst werden. Die Bearbeitungsdauer muss aber gleich sein wie bei der regulären Variante.

Spätmöglichster Starttermin: HS KW 38; FS KW 8

2. Regulärer Abgabetermin 03.01.2022

Individuelle Termine dürfen direkt angepasst werden. Die Bearbeitungsdauer muss aber gleich sein wie bei der regulären Variante.

BAA: Um zur ordentlichen Diplomierung im Sommer zugelassen zu werden, muss die Abgabe bis

| spätestens Freit | ag, eine Woche nach Semesterende, erfol | lgen. |
|---|--|---|
| 3. Studierende: | | |
| | Student/in 1: | Student/in 2: |
| Name, Vorname: | Kälin Mischa | n/a |
| Studiengang: | Bachelor Informatik | |
| Mobile: | +41 79 232 67 03 | - |
| E-Mail: | mischa.kaelin@stud.hslu.ch | |
| Projekt mit Arbeit- geber (bb-Studierende) | ☐ Ja ⊠ Nein | □ Ja □ Nein |
| 4. Auftraggeber/i | n (Rechnungsadresse) & Experte: | |
| | Auftraggeber/in | Bei Bachelorarbeiten: Vorschlag Experte* |
| Firma: | Smart-Up | |
| Ansprechperson: | | |
| Funktion: | | |
| Strasse: | | |
| PLZ / Ort: | | |
| Telefon: | | |
| E-Mail: | | |
| vorzuschlagen. Diese/r a | Vird kein Vorschlag erfasst, erfolgt die Z | udierenden haben und sollte nicht direkt im |
| 5. Betreuungsper | son: | |
| Martin Bättig, Dozent H | SLUI | |

6. Aufgabenstellung

| Titel: | Online-Charaktererstellung für «Das Schwarze Auge» |
|--|---|
| Ausgangslage und | In diesem Projekt soll eine Webapplikation erstellt werden, welche die |
| Problemstellung: | Charaktererstellung im Pen & Paper (P&P) «Das Schwarze Auge» (DSA) |
| The second secon | vereinfacht. Die Charaktererstellung bei P&P-Spielen ist oftmals äusserst |
| | komplex, aufwändig und im Falle von DSA zudem noch unübersichtlich. |
| | |
| | Die zu erstellende Webapplikation soll dem Benutzer eine übersichtliche |
| | effiziente Charaktererstellung ermöglichen und zusätzlich eine Erweiterbarkeit |
| | mit eigenen Ideen ermöglichen. Mit diesen Eigenschaften bietet die |
| | Webapplikation eindeutige Vorteile gegenüber bereits existierenden Tools, |
| | welche aus der Fangemeinde von DSA entstanden sind und i.d.R. eher |
| | unübersichtlich, ineffizient und fehleranfällig sind. |
| Ziel der Arbeit und | Eine vollständige Webapplikation zur Charaktererstellung in DSA mit folgenden |
| erwartete Resultate: | Teilzielen: |
| erwariete Kestitiate. | Erfassung der Requirements (verschiedene Quellen; z.B. bestehende Tools, |
| | eigene Erfahrungen oder auch Interviews mit Spielern). |
| | Konzept der Webapplikation basierend auf den Requirements. Dieses |
| | • |
| | beinhaltet Applikationsarchitektur sowie eine Untersuchung der rechtlichen |
| | Aspekte und Vertriebsmöglichkeiten (*). Implementation der Webannlikation unter Berücksichtigung der Techniken |
| | imprementation der wedappinktion tiller Derdenstelligting der rectalitier |
| | der modernen Softwareentwicklung (mindestens VCS, CI, Testkonzept mit |
| | Tests). |
| | Als akademischer Beitrag sollen mindestens drei verschiedene |
| | Technologieevaluationen durchgeführt werden (z.B. Architektur, Frontend- |
| | Technologie, Backend-Technologie, zentrales Coding-Pattern, DB, usw.). |
| | (*) Ziel ist die Bereitstellung des Produkts auf dem Skriptorium. Das Skriptorium |
| | ist eine Plattform, bereitgestellt von Ulisses (Macher von DSA), welches den |
| | Verkauf von spielrelevanten Tools, Abenteuern, usw. ermöglicht. Jedoch ist |
| | unklar, ob die Applikation direkt ausgeliefert wird oder lediglich Logins verkauft |
| | werden. |
| Gewünschte | Ob klassisch, hybrid oder agil bleibt dem Studierenden überlassen. Wichtig ist ein |
| Methoden, Vorgehen: | konsistentes und vollständig dokumentiertes Vorgehen (Nachvollziehbarkeit). |
| Kreativität, Varianten, | Das Projekt erlaubt Varianten und Innovation in verschiedenen Bereichen bspw. |
| Innovation* | durch ein ungewöhnliches UI, besondere Vertriebsmöglichkeit, automatische |
| | Vorschläge und Tipps, vereinfachter Modus, usw. |
| Sonstige | n/a |
| Bemerkungen: | |
| Wirtschaftsprojekt | ☐ Wirtschaftsprojekt: 180 Stunden pro Studierender |
| oder Bachelorarbeit: | ☐ Bachelorarbeit: 360 Stunden |
| | 23 Dachelolatoch. 300 Stulidell |

^{*} Bitte heben Sie in diesem Punkt hervor, inwiefern Ihre Projektidee über kreativen Spielraum verfügt. Dabei sind folgende Kriterien relevant: Die Idee erlaubt den Studierenden eigene Ideen zu entwickeln und Varianten zu erarbeiten, ist ausserhalb vom Tagesgeschäft angesiedelt, beinhaltet Neuland/Innovation und ist nicht durch Produkte & Tools getrieben.

Bitte kreuzen Sie eine Projektart und die zutreffenden Schwerpunkte an.

| Projektarten: | Schwerpunkte: |
|---|--|
| ☐ Einsatz von Standardsoftware und Services | ☐ Artificial Intelligence & Machine Learning |
| | ☐ Business Process Modelling |
| ☐ Innovationsprojekte (Projekte mit | ☐ Data Science |
| Erkenntnisgewinn, Forschungsprojekte) | ☐ Hardwarenahe Software-Erstellung |
| ☐ IT-Infrastrukturentwicklung | ☐ Human Computer Interaction Design |
| ☐ Strukturierte Analyse und Konzeption | ☐ ICT Business Solutions |
| | |

| Hochschule | Luzern |
|------------|--------|
| Informatik | |

| von | Systemen und Abläufen | | ☐ ICT Infrastrukt ☐ Internet of This ☐ Mobile System ☐ Security/Privac ☑ Software-Erste ☐ Visual Comput | ngs s ry llung |
|-----|--|-----------------------------|---|--|
| 7. | Zeiteinteilung Vorschlag für die Zeitein | teilung <u>pro Person</u> | Bildverarbeitur | ing (Grank, ng, Vision, VR, AR) |
| | WIPRO: pro Woche: für Modulendprüfung: Total: | ca. 12h ca. 10h 180 h | BAA: pro Woche: Schlusswoche: Für Modulendprüfung: Total: | ca. 20h ca. 50h ca. 10h 360 h |

8. Rechtliche Grundlagen und Reglemente

Folgende Rechtsgrundlagen und Reglemente sind für die Wirtschaftsprojekte und Bachelorarbeiten an der Hochschule Luzern – Informatik massgebend:

- Studienordnung für die Ausbildung an der Hochschule Luzern, FH Zentralschweiz (Link)
- Studienreglement für die Bachelor-Ausbildung an der Hochschule Luzern Informatik (Link)

9. Bestätigung

Mit der Kenntnisnahme der Aufgabenstellung bestätigen Student/in und Auftraggeber/in, dass

- Sie mit der Aufgabenstellung einverstanden sind.
- die Auftraggeberin/der Auftraggeber damit einverstanden ist, dass die Hochschule Luzern Informatik für die Organisation einer <u>Bachelorarbeit</u> von ihr/ihm einen Kostenbeitrag von CHF 1'000.00 (inkl. MwSt.) pro Student/in erhebt. Dies gilt nicht für Arbeiten, welche berufsbegleitend Studierende in Verbindung mit ihrem Arbeitgeber/ihrer Arbeitgeberin machen und für HSLU interne Auftraggeber/innen. Für die Wirtschaftsprojekte wird kein Kostenbeitrag verrechnet.
- Betreuungspersonen und Experten uneingeschränkten Einblick in die Arbeit erhalten. Auch anlässlich von Präsentationen und Marketingaktivitäten kann die Arbeit der Öffentlichkeit gezeigt werden. Falls die Arbeit vertraulich behandelt werden soll, muss der Aufgabenstellung eine entsprechende Vertraulichkeitserklärung beiliegen. Eine Zusammenfassung der Arbeit wird in jedem Fall veröffentlicht.

| Datum: | 27.9.2021 | |
|--------|-----------|--|
|--------|-----------|--|

Die definitive Aufgabenstellung (pdf-Format) bitte per E-Mail an die Transferstelle senden, zwingend in Kopie an alle involvierten Parteien.

Anlaufstelle für alle Informationen im Zusammenhang mit studentischen Arbeiten sowie für Entgegennahme von Projektideen & Aufgabenstellungen:

Hochschule Luzern - Informatik Transfer Services Suurstoffi 1 6343 Rotkreuz T: 041 228 24 66

E: transfer.informatik@hslu.ch

39 | 63

8.3 Projektmanagement

In diesem Kapitel befinden sich das Backlog, wie in der Tabelle 8 zu sehen, und die fünf Protokolle zu den jeweiligen Sprints inklusive Retrospektive und Burndown-Chart.

Tabelle 8: Backlog

| Sprint | Name | Label | | | |
|--------|---|----------|--|--|--|
| | Projekt inklusive CI aufsetze | Code | | | |
| | Installation von UI Framework | Code | | | |
| | Software-Architektur festlegen | Research | | | |
| 4 | InitPage, Layout und Farben definieren | | | | |
| 1 | Charakter mit Particulars und Attributes initialisieren | Code | | | |
| | Advantages & Disadvantages hinzufügen | Code | | | |
| | Skills hinzufügen | Code | | | |
| | Export des Charakter-Sheets | Code | | | |
| | Testkonzept definieren | Research | | | |
| | Anforderungen ausschreiben | Research | | | |
| 2 | Geburtstag mit Tag und Monat | Code | | | |
| | Inventory initialisieren | Code | | | |
| | GeneralSpecialAbility hinzufügen | Code | | | |
| | CombatTechnique hinzufügen* | Code | | | |
| | Kampf-Tab initialisieren* | Code | | | |
| 3 | Laden des Charakters | Code | | | |
| | Ausweichen und Initiative hinzufügen | Code | | | |
| | Abgeleitete Charaktereigenschaften hinzufügen | Code | | | |
| | Konzept und Realisation der Validierung* | Research | | | |
| | Caching-Strategie für Projekt identifizieren | Research | | | |
| 4 | Gesamtkosten des Charakters soll berechnet werden | Code | | | |
| | Validation soll mit Tests ausgestattet werden | Code | | | |
| | Konzept für Evaluation und Validation | Research | | | |
| | Umsetzung als PWA | Code | | | |
| 5 | Validation & Evaluation erarbeiten | Research | | | |
| J | Ausblick dokumentieren | Research | | | |
| | Abstract & Web-Abstract verfassen | Research | | | |

^{*} bedeutet, dass die Arbeitspakete ursprünglich im vorhergehenden Sprint eingeplant waren.

8.3.1 Sprint #1

Das Produkt nimmt Gestalt an und wirkt robust. Der erste Eindruck ist zufriedenstellend und dient als Grundlage für Diskussionen bezüglich neuer Komponenten und Verbesserungspotenzial.

Retrospektive

Trotz Initialisierungsphase war dieser Sprint sehr chaotisch: Die Backlog-Items wurden nicht nacheinander gelöst, sondern aufgrund ungeplanter Abhängigkeiten vorgezogen und dadurch parallel abgearbeitet. Dazu kommen die unpräzisen Schätzungen, weil oftmals frühere Komponenten angepasst werden mussten.

Mehrfach wurde im Verlauf des Sprints Zeit für Refactoring investiert, was sich bereits hier ausgezahlt hat. Diese Zeit für das Code-Refactoring sollte für die folgenden Sprints nicht vernachlässigt werden. Trotzdem konnte der Sprint zeitnah abgeschlossen werden.

Burndown-Chart

Die Abbildung 15 zeigt das Burndown-Chart des ersten Sprints vom 25. Oktober bis 7. November.



Abbildung 15: Sprint 1 - Burndown-Chart

8.3.2 Sprint #2

Das Produkt wurde in diesem Sprint verfeinert und mit einem ersten Konzept für ein Inventar ergänzt. Mit Hilfe von Unit-Tests konnte das Datenmodell korrigiert werden.

Retrospektive

Weil etwas mehr Zeit als erwartet für das Testkonzept benötigt wurde, konnten nicht alle Arbeitspakete erledigt werden. Dafür konnten die nachgereichten Tests noch fehlerhafte Berechnungen im Datenmodell aufzeigen, was den Verzug rechtfertigt. Die übrig gebliebenen Arbeitspakete werden im nächsten Sprint mit hoher Priorität untergebracht, weil sie Teil der Zwischenpräsentation sind.

Ähnlich wie beim ersten Sprint wurden Teile des Projekts refaktorisiert, was dieses Mal kaum Mehraufwand verursacht hat. Trotzdem zeigt dies die Wichtigkeit von sauberen, modularen Projekten, um Erweiterungen und Bugfixes vorzunehmen.

Im Gegensatz zum ersten Sprint hatte ich nicht mehr die Möglichkeit, mehr Aufwand als geplant in den Sprint zu investieren. Vermutlich habe ich mich daher zusätzlich noch überschätzt.

Burndown-Chart

Die Abbildung 16 zeigt das Burndown-Chart des zweiten Sprints vom 8. November bis 21. November.

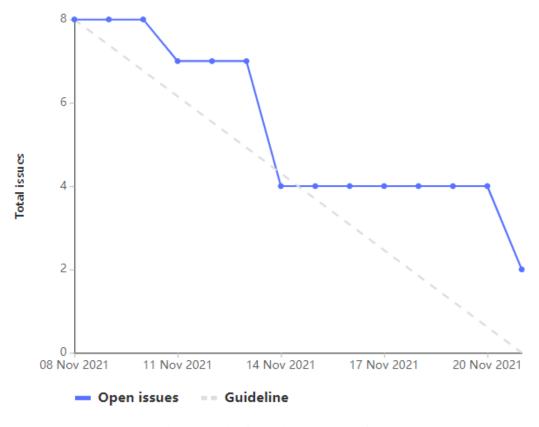


Abbildung 16: Sprint 2 - Burndown-Chart

8.3.3 Sprint #3

Zu Anfang des Sprints wurden noch Tickets aus dem vorhergehenden Sprint bearbeitet, damit diese schon bei der Zwischenpräsentation vorhanden sind. Nebst diesen Abschlussarbeiten stand die Validierung des Charakters im Vordergrund.

Retrospektive

Die Aufteilung der Items basierend auf der Aufwandschätzungen war etwas unglücklich. Einerseits waren sehr kleine Arbeitspakete vorhanden, welche innerhalb kurzer Zeit abgeschlossen wurden. Dabei habe ich den Gesamtaufwand des Sprints leicht unterschätzt und immer noch etwas CSS und HTML überarbeitet. Dies wurde mir gegen Ende des Sprints zum Verhängnis, als nur noch die aufwändige Validierung offen war. Obwohl ich die technische Implementation dieser fertigstellen konnte, hatte ich nicht mehr genügend Zeit für die Dokumentation.

Eine bessere Reihenfolge oder auch voraussichtliche Planung hätte dieses Problem allenfalls verhindern können. In Zukunft versuche ich, vermehrt das Sprintziel zu fokussieren und nicht zusätzliche Arbeiten um die Arbeitspakete herum zu realisieren. Ein anderer Ansatz wäre die Aufteilung von Implementation und der dazugehörigen Dokumentation in zwei Arbeitspakete, wodurch auch die Schätzung erleichtert wird.

Burndown-Chart

Die Abbildung 17 zeigt das Burndown-Chart des dritten Sprints vom 22. November bis 5. Dezember.



Abbildung 17: Sprint 3 - Burndown-Chart

8.3.4 Sprint #4

Innerhalb dieses Sprints wurde die Applikation grundsätzlich fertiggestellt, damit im folgenden Sprint die Evaluation und Validation durchgeführt werden kann. Einzig noch offen ist die Realisation als PWA.

Retrospektive

Der Sprint war für mich äusserst zufriedenstellend. Ich konnte einige Arbeiten abschliessen und die letzten Konzepte für den letzten Sprint definieren. Hinzukommt die ausgewogene Erledigung der Arbeitspakete, was für einen ausgeglichenen Sprint gesorgt hat. Die Abwechslung zwischen Code und Dokumentation war auch sehr passend.

Leider war das letzte Wochenende noch etwas chaotisch: Einerseits habe ich die Vollendung des letzten Arbeitspakets nicht am Samstag vermerkt. Andererseits sorgten am Sonntag noch private Gründe für einen verspäteten Abschluss des Sprints, obwohl eigentlich alle Arbeitspakete pünktlich abgeschlossen wurden.

Gerne würde ich mich in Zukunft wieder sorgfältiger mit der agilen Planung befassen, damit auch keine überlappenden Sprints mehr aufkommen.

Burndown-Chart

Die Abbildung 18 zeigt das Burndown-Chart des vierten Sprints vom 6. Dezember bis 20. Dezember. Aus privaten Gründen musste der Abschluss des Sprints um einen Tag verschoben werden.



Abbildung 18: Sprint 4 - Burndown-Chart

8.3.5 Sprint #5

Als abschliessender Sprint dieser Arbeit wurde dem Produkt ein Service-Worker hinzugefügt, wodurch die Lauffähigkeit offline garantiert wird. Zudem wurde das Endprodukt durch die eigene Spielgruppe bewertet und die Dokumentation komplettiert.

Retrospektive

Der zeitliche Aufwand dieses Sprints wurde bewusst tiefer geschätzt als bei vorhergehenden Sprints. Grund dafür sind einerseits die Festtage und andererseits der Abgabetermin. Schlussendlich müssen Projekt und Dokumentation noch hochgeladen werden, weshalb nicht bis am letzten Tag des Sprints gearbeitet werden kann. Diese Strategie ist aufgegangen und der Sprint konnte zeitgerecht abgeschlossen werden.

Burndown-Chart

Die Abbildung 19 zeigt das Burndown-Chart des fünften Sprints vom 20. Dezember bis 2. Januar.

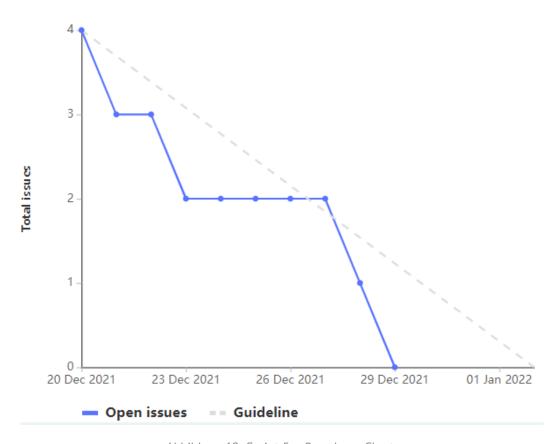


Abbildung 19: Sprint 5 - Burndown-Chart

8.4 Testdrehbücher

Die Testdrehbücher dienen zur Überprüfung der Qualitätskontrolle und ergänzen die automatischen Tests, welche durch die CI ausgeführt werden. In der Regel handelt es sich dabei um umfängliche Systemtests mit einer definierten Ausgangslage, mehreren Zwischenschritten sowie mindestens einem erwarteten Resultat.

8.4.1 Testprotokoll

Die Tabelle 9 zeigt die Resultate der manuell ausgeführten Tests in einem zeitlichen Verhältnis.

Tabelle 9: Testprotokoll

| | 29.11 | 02.11 | 03.12 | 05.12 | 08.12 | 09.12 | 19.12 | 20.12 | 26.12 |
|-------------------|-------|-------|-------|-------|-------|--------------|-------|-------|-------|
| Import/Export | ✓ | × | × | × ✓ | ✓ | \checkmark | × | ✓ ✓ | ✓ |
| Validation | - | - | - | - | × | ✓ | × | × ✓ | ✓ |
| Google Lighthouse | - | - | - | - | - | - | - | - | ✓ |

8.4.2 Import/Export

Ziel dieses Tests ist die Überprüfung, ob alle Inhalte korrekt importiert exportiert werden. Dieser Test wird für jeden vorhandenen Charakterbogen, der für Testzwecke benötigt wird, durchgeführt.

- 1. Applikation starten
- 2. Charakterbogen mit «Laden...» öffnen; prüfen, ob einige Felder abgefüllt sind
- 3. Datei mit «Export» erstellen; prüfen, ob die exportierte und importierte Datei identisch sind

8.4.3 Validation

Bei diesem Test liegt der Fokus auf dem Inhalt des Tabs «Log» und dessen Aktualisierung bei Änderungen.

- 1. Applikation starten
- 2. «DummySheet_1.json» mit «Laden...» öffnen; prüfen, ob exakt 7 Einträge im «Log» enthalten sind
- 3. Magiekunde unter Talente auf 4 erhöhen; prüfen, ob exakt 6 Einträge im «Log» enthalten sind
- 4. Eigenschaften IN und GE auf 15 erhöhen; prüfen, ob exakt 4 Einträge im «Log» enthalten sind
- 5. Ballistischer Schuss unter Kampf entfernen; prüfen, ob exakt 2 Einträge im «Log» enthalten sind
- 6. Eigenschaft MU auf 15 erhöhen; prüfen, ob exakt 3 Einträge im «Log» enthalten sind

8.4.4 Google Lighthouse

Für die Prüfung mit Google Lighthouse wird am besten das gleichnamige Add-On im Browser installiert und für die Applikation ausgeführt.

8.5 Inhaltsrichtlinie für Scriptorium Aventuris

Jeff Montgomery – 5. April 2021 21:37

8.5.1 Regeln

Du kannst alle Regeln des Spielsystems aus dem von Ulisses Spiele veröffentlichten "Das Schwarze Auge – Regel-Wiki" verwenden. Die Verwendung anderer offizieller Texte gleich welcher Art ist explizit nicht gestattet.

Du hast die Erlaubnis, die Namen Das Schwarze Auge, Aventuria, Dere, Myranor, Riesland, Tharun und Uthuria in deinem Werk zu verwenden. Darüber hinaus ist es dir gestattet, die Regeln in jedes andere offene Lizenzsystem (wie z.B. FATE, Open D6 usw.) oder selbst entwickelte System zu konvertieren, vorausgesetzt dein Werk verletzt nicht die Bedingungen der Lizenz des Zielsystems.

8.5.2 Settings

Du kannst Inhalte für Dere, das Setting von Das Schwarze Auge, erstellen, einschließlich der Kontinente Aventuria, Myranor, Riesland, Tharun und Uthuria. Ferner kannst du alternative Welten oder Historien und Produkte erstellen, die vergangene oder zukünftige Ereignisse behandeln.

8.5.3 Erlaubte Medien

Rollenspielergänzungen, Spielwelterweiterungen, Abenteuer, Literatur, Karten, Landkarten und Illustrationen sind unter dieser Lizenz allesamt erlaubt, ebenso Audioproduktionen und digitale Spiele. Videoproduktionen sind bis auf weiteres nicht erlaubt.

8.5.4 Illustrationen

Es dürfen keine Illustrationen oder grafischen Gestaltungselemente aus Publikationen von Ulisses Spiele verwendet werden, mit Ausnahme jener Illustrationen und grafischen Gestaltungselemente, die im Rahmen der SCRIPTORIUM-AVENTURIS-Illustrationspakete zur Verfügung gestellt werden. Du darfst Illustrationen und grafische Gestaltungselemente aus der Stock-Art-Sammlung von DriveThruRPG, aus der Public Domain oder von dir selbst erstellte oder in Auftrag gegebene Originalillustrationen und grafischen Gestaltungselemente verwenden. In jedem Fall musst du die Erlaubnis haben, die jeweilige Illustration oder das jeweilige grafische Gestaltungselement zu verwenden.

8.5.5 Logos und Handelsaufmachung

Das SCRIPTORIUM-AVENTURIS-Logo muss auf dem Deckblatt deines Werks erscheinen. Es muss bei Betrachtung in voller Größe lesbar sein. Es muss und sollte nicht zwangsläufig das vorrangige Logo oder die vorrangige Titelgrafik deines Deckblatts sein. Du darfst keine anderen Logos oder Handelsaufmachungen von Publikationen von Ulisses Spiele verwenden, mit Ausnahme dessen, was im Rahmen der SCRIPTORIUM-AVENTURIS-Illustrationspakete von Ulisses Spiele zur Verfügung gestellt wird.

8.5.6 Erstellungsvorgaben

Wir empfehlen dir, zur Erstellung deines Werks die SCRIPTORIUM-AVENTURIS-Vorlage zu verwenden, wozu du jedoch nicht verpflichtet bist.

8.5.7 Rechtliches

Der folgende Text muss stets im Werk enthalten sein (und ist dort in das Werk einzufügen, wo alle anderen rechtlichen und Copyright-Hinweise aufgeführt werden):

Dieses Produkt wurde unter Lizenz erstellt. Das Schwarze Auge und sein Logo sowie Aventuria, Dere, Myranor, Riesland, Tharun, Uthuria, The Dark Eye und ihre Logos sind eingetragene Marken von Ulisses Medien und Spiele Distribution GmbH in Deutschland, den U.S.A. und anderen Ländern. Ulisses Spiele und sein Logo sind eingetragene Marken der Ulisses Medien und Spiele Distribution GmbH.

Dieses Werk enthält Material, das durch Ulisses Spiele und/oder andere Autoren urheberrechtlich geschützt ist. Solches Material wird mit Erlaubnis im Rahmen der Vereinbarung über Gemeinschaftsinhalte für SCRIPTORIUM AVENTURIS verwendet.

Alle anderen Originalmaterialien in diesem Werk sind Copyright [2016] von [dein Name oder der deiner Firma] und werden im Rahmen der Vereinbarung über Gemeinschaftsinhalte für SCRIPTORIUM AVENTURIS veröffentlicht.

8.5.8 Sonstiges

Weder dein Werk noch sonstiges Werbematerial, einschließlich Blogeinträgen oder Pressemeldungen, dürfen rassistische, homophobe, diskriminierende oder anstößige Ansichten, offenkundig politische Agenden oder Ansichten, Darstellungen oder Beschreibungen krimineller Handlungen gegenüber Kindern, Vergewaltigung oder andere Akte krimineller Perversion oder sonstiges obszönes Material ohne ausdrückliche schriftliche Erlaubnis durch Ulisses Spiele in Form eines von dieser Lizenz unabhängigen Dokuments enthalten.

Illegale und verletzende Inhalte sind nicht gestattet. Es obliegt der Verantwortung des Schöpfers des Inhaltes, sicherzustellen, dass seine oder ihre Inhalte nicht gegen Gesetze, Urheberrechte, Markenrechte, Privatsphäre oder sonstige Rechte verstoßen.

8.6 Anforderungen

Dieses Kapitel umfasst die Auflistung der funktionalen Anforderungen, sortiert nach absteigender Priorität Die Anforderungen werden anhand eines einzigartigen Identifikators, einer Beschreibung, der Quelle und der Volatilität (hoch, mittel, tief) beschrieben. Letzteres beschreibt das Risiko, dass sich die Anforderung noch während des Entwicklungsprozesses ändert.

8.6.1 Charaktererstellung und -manipulation

| Identifikator | #01 |
|---------------|---|
| Beschreibung | Die Software muss dem User ermöglichen, einen neuen Charakter für DAS zu erstellen oder einen bestehenden zu bearbeiten. Dabei müssen mindestens alle regeltechnischen Eigenschaften ausfüllbar sein. Zusätzliche Angaben zur Person (Name, Geschlecht, Geburtstag) müssen ebenfalls erfassbar sein. |
| Quelle | Auftraggeber |
| Volatilität | Tief |

8.6.2 Einhaltung aller rechtlichen Aspekte

| Identifikator | #02 |
|---------------|---|
| Beschreibung | Alle rechtlichen Aspekte müssen eingehalten werden. Dies bezieht sich auf die Verwendung des geistigen Eigentums von Ulisses sowie die erlaubte Verwendung von Frameworks, Entwicklungsumgebungen, etc. |
| Quelle | Auftraggeber |
| Volatilität | Tief |

8.6.3 Persistenz des Charakters

| Identifikator | #03 |
|---------------|--|
| Beschreibung | Die Persistenz des Charakters muss gewährleistet sein. Das heisst, der Charakter muss gespeichert und zu einem anderen Zeitpunkt wieder identisch geladen werden können. |
| Quelle | Ulisses |
| Volatilität | Tief |

8.6.4 Keine lokale Installation notwendig

| Identifikator | #04 |
|---------------|---|
| Beschreibung | Die Software muss ohne zusätzlich Installation ausführbar sein. Dies beinhaltet nicht die Installation eines Browsers oder einer Software zur Darstellung von Dokumenten wie PDF. |
| Quelle | Auftraggeber |
| Volatilität | Tief |

8.6.5 Lauffähig offline

| Identifikator | #05 |
|---------------|--|
| Beschreibung | Die Software muss auch offline zur Verfügung stehen und mindestens das Laden eines Charakters erlauben. Ein initialer Austausch mit einem Server zum Laden der Stammdaten ist möglich, insofern diese Verbindung danach nicht mehr notwendig ist. Bestenfalls wird die Verbindung mit einem Server nur für Updates der Software (inklusive Stammdaten) benötigt. |
| Quelle | Mitspieler |
| Volatilität | Tief |

8.6.6 Berechnungen sind korrekt und komplett

| Identifikator | #06 |
|---------------|--|
| Beschreibung | Alle Berechnungen innerhalb des Charakterbogens müssen visuell ausgewiesen und korrekt berechnet werden, um diese mit der Erfahrung des Charakters zu vergleichen. |
| Quelle | Auftraggeber |
| Volatilität | Tief |

8.6.7 Validierung der Abhängigkeiten (Integrität des Charakters)

| Identifikator | #08 |
|---------------|---|
| Beschreibung | Die Voraussetzungen von ausgewählten Komponenten müssen für den Charakter überprüft werden, um dessen Integrität zu wahren. Es müssen stets alle verfügbaren Komponenten zur Verfügung stehen, jedoch sollen nicht erfüllte Voraussetzungen dem Benutzer mitgeteilt werden. Eine Speicherung soll in jedem Fall möglich sein. |
| Quelle | Auftraggeber |
| Volatilität | Mittel |

8.6.8 Manuelle Erweiterbarkeit zur Verfügung stellen

| Identifikator | #07 |
|---------------|---|
| Beschreibung | Die Software muss manuelle Anpassungen zu den Stammdaten erlauben, wobei dieser Prozess anfangs nicht benutzerfreundlich sein muss. |
| Quelle | Mitspieler |
| Volatilität | Mittel |

8.6.9 Standardisierter Json-Export als Schnittstelle

| Identifikator | #09 |
|---------------|---|
| Beschreibung | Für den einfachen Austausch zwischen anderen Tools wie Optolith, Foundry VTT und Das Weisse Auge soll ein standardisierter Json-Export zur Verfügung gestellt werden. |
| Quelle | Mitspieler |
| Volatilität | Hoch (neue Version 2.0 derzeit in Arbeit) |

8.6.10 Standardisierter Json-Import als Schnittstelle

| Identifikator | #10 |
|---------------|---|
| Beschreibung | Für den einfachen Austausch zwischen anderen Tools wie Optolith, Foundry VTT und Das Weisse Auge soll ein standardisierter Json-Import zur Verfügung gestellt werden. |
| Quelle | Mitspieler |
| Volatilität | Hoch (neue Version 2.0 derzeit in Arbeit) |

8.7 Fragebogen

Jegliche von den Teilnehmenden bereitgestellten persönlichen Informationen werden streng vertraulich behandelt. Durch die Teilnahme an dieser Umfrage stimmst du der Verwendung deiner Antworten in aggregierter Form für die Weiterentwicklung der Applikation und ohne das Recht einer Gegenleistung zu.

| | Nein | Eher nein | Eher ja | Ja |
|---|--------------------|-----------|---------|------------|
| Ist die Applikation für dich übersichtlich gestaltet? | | | | |
| Empfindest du die Aufteilung der Inhalte als sinnvoll? | | | | |
| Findest du die Applikation selbsterklärend? | | | | |
| Hat die Applikation deine Anpassungen ohne drastische Verzögerung übernommen? | | | | |
| Würdest du dieses Produkt nach Release verwenden? | | | | |
| Bewerte folgenden Funktionalitäten nach ihrer Wichtigk | eit: | | | |
| Bewerte folgenden Funktionalitäten nach ihrer Wichtigk | eit: | | | |
| Bewerte folgenden Funktionalitäten nach ihrer Wichtigk | eit: Irrelevant | Unwichtig | Wichtig | Essenziell |
| Bewerte folgenden Funktionalitäten nach ihrer Wichtigke Manuelle Erfassung von eigenen Inhalten | | Unwichtig | Wichtig | Essenziell |
| | Irrelevant | | | |
| Manuelle Erfassung von eigenen Inhalten | Irrelevant | | | |
| Manuelle Erfassung von eigenen Inhalten Standardisierter Json-Export als Schnittstelle | Irrelevant | | | |
| Manuelle Erfassung von eigenen Inhalten Standardisierter Json-Export als Schnittstelle Standardisierter Json-Import als Schnittstelle | Irrelevant | | | |

Traten Crashes, Anzeigefehler oder anderweitige Probleme bei der Nutzung der Applikation auf?

Vielen Dank für deine Teilnahme!

Mischa Kälin

Bemerkungen

8.7.1 Resultate

Die Resultate beruhen auf den Antworten der fünf Mitspieler aus der Spielgruppe. Die Fragen sind analog zum Fragebogen dargestellt mit dem Unterschied, dass die Kontrollkästchen durch die Anzahl der jeweils erhaltenen Antwort ersetzt wurden. Für die offenen Fragen wurde eine Liste mit den Antworten erstellt.

| | Nein | Eher nein | Eher ja | Ja |
|---|------|-----------|---------|----|
| Ist die Applikation für dich übersichtlich gestaltet? | 0 | 0 | 4 | 1 |
| Empfindest du die Aufteilung der Inhalte als sinnvoll? | 0 | 0 | 2 | 3 |
| Findest du die Applikation selbsterklärend? | 3 | 0 | 2 | 0 |
| Hat die Applikation deine Anpassungen ohne drastische Verzögerung übernommen? | 0 | 0 | 1 | 4 |
| Würdest du dieses Produkt nach Release verwenden? | 1 | 1 | 1 | 2 |

| | Irrelevant | Unwichtig | Wichtig | Essenziell |
|--|------------|-----------|---------|------------|
| Manuelle Erfassung von eigenen Inhalten | 0 | 0 | 2 | 3 |
| Standardisierter Json-Export als Schnittstelle | 0 | 0 | 2 | 3 |
| Standardisierter Json-Import als Schnittstelle | 0 | 3 | 0 | 2 |
| Spielunterstützung mit Schaden, Status, etc. | 2 | 3 | 0 | 0 |
| Möglichkeit direkt in der Applikation zu würfeln | 2 | 3 | 0 | 0 |

Was vermisst du in der Applikation sonst noch?

- Wiki, Weiterleitung auf Ulisses Wiki bei Fähigkeiten, Beschreibung der Fähigkeiten, Hilfestellung bei Erstellung. Hinweis auf Abhängigkeiten und Voraussetzungen, Professionspakete
- Print-Version für A4
- Funktionierendes Inventar
- Inventar (Möglichkeit, persönliche Gegenstände und Waffen hinzuzufügen, Gewicht / Belastung anzeigen, Geldverwaltung etc.)
- Vollständige Auswahlmöglichkeiten (z.B. Magie, Sonderfertigkeiten)
- Möglichkeit, Worddokument / PDF zu generieren, mit allen Beschreibungen zu Sonderfertigkeiten / Zaubern / etc.

Traten Crashes, Anzeigefehler oder anderweitige Probleme bei der Nutzung der Applikation auf?

- Ich will nicht das mein Charakter jedes Mal gelöscht wird, wenn ich auf home klicke.
- Wenn ich zu viele Punkte verwende kommt die Fehlermeldung im Log dafür 2mal (Tritt nur auf wenn ich mit einem geladenen Charakter arbeite, nicht bei einem neuen)
- Die Zahlfelder haben einen Anzeigefehler, wenn ich einen nicht erlaubten Wert eingebe wird dieser nur beim ersten Versuch nicht akzeptiert, wenn man nochmals einen falschen Wert eingibt wird dieser akzeptiert. Der Fehler scheint rein visuell zu sein.

Bemerkungen

- Das GUI erfüllt seinen Zweck, wird aber keine Kunden Bindung erzeugen. Zu steril, eigener touch fehlt. Sehr rudimentär.
- Ich würde gerne meine eigenschaftswerte/Leben/AsP/KaP immer ausserhalb der Reiter sehen können. Wieso gleiches Element in 3 reitern anzeigen. Auch falls der Wunsch des Entwicklers ist dieses Programm als PDF zu ersetzen sollte man die Wichtigstens Sachen immer angezeigt haben.
- Inventar Reiter zwar vorhanden aber ohne Funktionalität. Wirkt so als sollte man es testen, aber es wird schnell klar das dies noch gar nicht dazu gehört. Empfehlung ist nur den Tester mit Sachen konfrontieren welche er auf bis zum ausreizen testen kann. Das gleiche bei der Rüstkammer.
- Die letzten beiden Punkte (Spielunterstützung mit Schaden, Status, etc. & Möglichkeit direkt in der Applikation zu würfeln) sind abhängig von dem Umfang der Nutzung der Seite. Wird die Seite nur zur Erstellung verwendet, oder das json für eine andere Applikation (z.B. <u>foundry.aven-turicum.ch</u>) exportiert, sind sie unnötig, ansonsten essenziell.
- Bei Kampf: Initiative, Ausweichen, Geschwindigkeit sind blau hinterlegt, obwohl nicht manuell änderbar.
- Bei Talenten: Funktion der Checkboxes nicht klar.

8.8 Google Lighthouse: Verbesserungsvorschläge

Die Verbesserungsvorschläge beruhen auf der Überprüfung des Endprodukts durch Google Lighthouse.

8.8.1 Performance

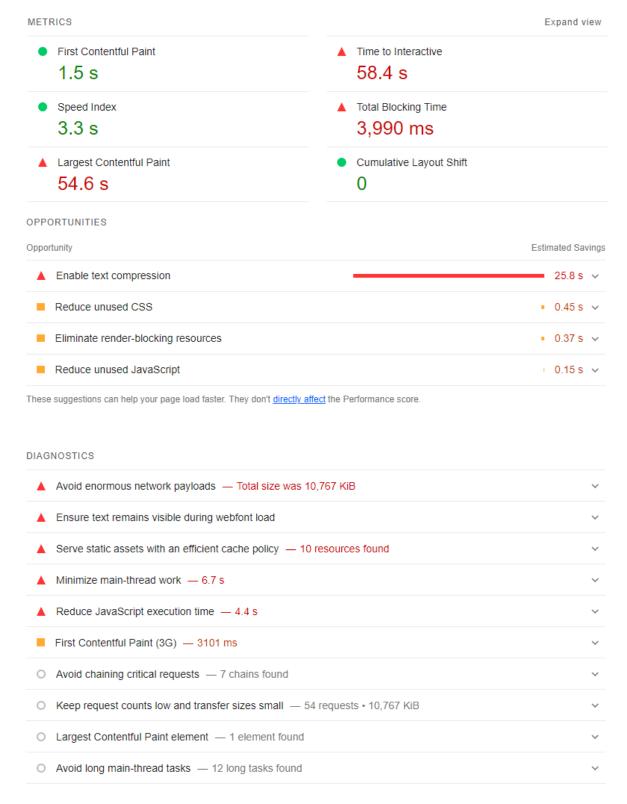


Abbildung 20: Google Lighthouse - Performance

8.8.2 Accessibility

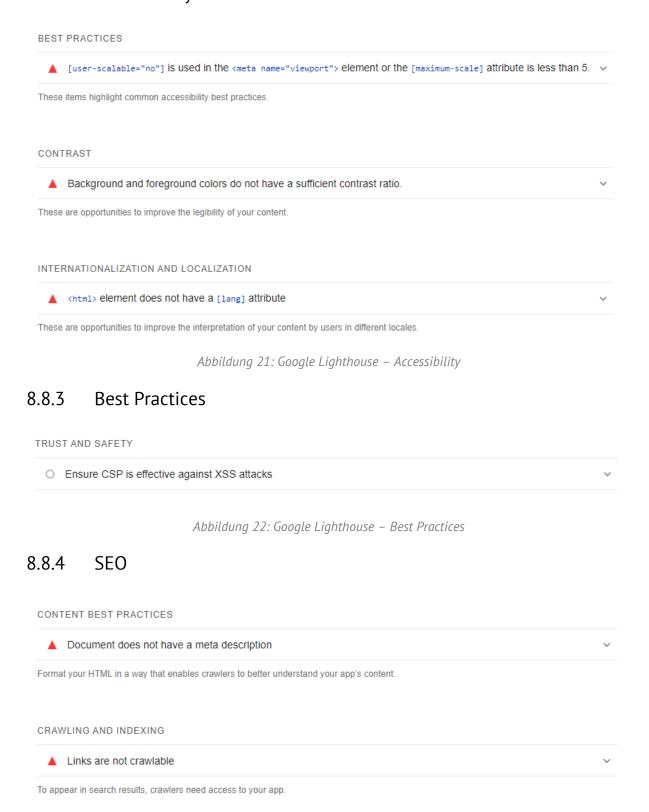


Abbildung 23: Google Lighthouse - SEO

8.8.5 PWA

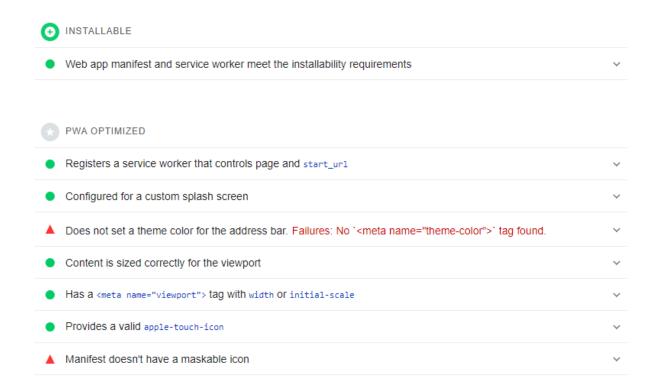


Abbildung 24: Google Lighthouse - PWA

9 Verzeichnisse

9.1 Abkürzungsverzeichnis

- CI Der DevOps-Prozess Continuous Integration (kurz CI) beschreibt die automatische Ausführung von Build und Tests, sobald Codeänderungen auf dem zentralen Repository registriert werden.
- **D&D** «Dungeon & Dragons» (kurz D&D) ist das bekannteste Pen-&-Paper-RPG. (Beresford 2011)
- **DoD** Definition-of-Done (kurz DoD) steht im Scrum für die allgemeinen Anforderungen an die Arbeitspakete, bevor diese abgeschlossen werden können (Madan 2019).
- **DSA** «Das Schwarze Auge» (kurz DSA) ist ein deutsches Pen-&-Paper-RPG.
- NPC Non-Player Character (kurz NPC) beschreibt all jene Charaktere in Spielen, die nicht von einem der Protagonisten kontrolliert wird (Montelli 2021). In Pen-&-Paper-RPGs zählt der Gamemaster nicht als Protagonist, weil dieser nicht im Interesse des jeweiligen Charakters, sondern der Geschichte handelt.
- **PWA** Progressive Web-Applikationen (kurz PWA) sind Applikationen, mit erweiterten Möglichkeiten: Wie traditionelle Web-Applikationen basieren auch PWAs auf HTML, CSS, JavaScript und/oder WebAssembly. Darüber hinaus können PWAs, sofern erlaubt, auf native Funktionen wie Kamera oder Mikrofon des Geräts zugreifen. (Toonen 2020)
- RPG Als Genre ist Role Playing Game (kurz RPG) häufig bei Videospielen, aber auch analogen Spielen, anzutreffen. In der Regel beinhalten solche Spiele allermeist eine Charakterentwicklung, Kampferfahrung (abhängig von den Charaktereigenschaften) und ein Inventar. (Stegner 2020)
- **SEO** Die Search Engine Optimization (kurz SEO) ist der Prozess zur Verbesserung einer Webseite bei organischen, also nicht bezahlten, Suchresultaten. (Moz 2021)
- SoDa Als Vorgehensmodell eignet sich Software Development agile (kurz SoDa) besonders für Software-Entwicklungsvorhaben mit kleineren Teams über einige Monate. Aber auch schulische Projektarbeiten sind möglich zu realisieren, weshalb auch SoDa tendenziell dokumentenlastig ist. Nicht alle Artefakte müssen für jedes wirtschaftliche Projekt nötig sein. (Jud, et al., 2020)
- UI User Interface (kurz UI) beschreibt die interaktive Schnittstelle zwischen Mensch und einer
 Maschine in Form eines Computer, Website oder Applikation. (Indeed Editorial Team 2021)
- **W3C** Das World Wide Web Consortium (kurz W3C) ist eine internationale Organisation zur Standardisierung von Technologien im Bereich Web. (W3C 2021)

9.2 Abbildungsverzeichnis

| Abbildung 1: Screenshot aus dem DSA5 Deluxe Heldendokument | 2 | | |
|--|----------------------|--|----|
| Abbildung 2: Screenshot aus Foundry VTT | 2 | | |
| Abbildung 3: Screenshot aus dem Optolith Character Generator | 3 | | |
| Abbildung 4: Screenshot aus The Dark Aid | 3 | | |
| Abbildung 5: Domänenmodell für Charakter in DSA (Grundregelwerk) | | | |
| | | Abbildung 7: Unterschied von Agil und Wasserfall (Amsterdam Standard 2021) | 9 |
| Abbildung 8: Schichtenmodell für Client und Server | 14 | | |
| Abbildung 9: Network only (Theo 2021) | 17 | | |
| Abbildung 10: Network first (Theo 2021) | 18 | | |
| Abbildung 11: Stale-While-Revalidate (Theo 2021) | 18 | | |
| Abbildung 12: Multi-Level Chained Binding | 24 | | |
| Abbildung 13: Aufteilung der Anforderungen | 26 | | |
| Abbildung 14: Ergebnisse von Google Lighthouse | 31 | | |
| Abbildung 15: Sprint 1 – Burndown-Chart | 42 43 55 56 | | |
| | | Abbildung 22: Google Lighthouse – SEO | 56 |
| | | Abbildung 23: Google Lighthouse – PWA | 57 |
| | | | |
| | | | |
| | | 9.3 Codeverzeichnis | |
| | | Code 1: Manifest.json | 5 |
| Code 2: Liste von Items als Json | 23 | | |
| Code 3: Two-Way-Binding in HTML – Parent | 24 | | |
| Code 4: Two-Way-Binding in HTML – Child | 24 | | |
| Code 5: Two-Way-Binding in HTML – Parent (angepasst) | 25 | | |
| Code 6: Property AttributesWrapper | 25 | | |
| | | | |

9.4 Tabellenverzeichnis

| Tabelle 1: Komponenten und deren Anforderungen | 20 |
|--|----|
| Tabelle 2: UI Frameworks und deren Abdeckung von Anforderungen | 21 |
| Tabelle 3: Resultat der Fragen zu den nicht-funktionalen Anforderungen | 28 |
| Tabelle 4: Resultat der Fragen zu den funktionalen Anforderungen | 29 |
| Tabelle 5: Übersicht der Anforderungen und deren Verifizierung | 30 |
| Tabelle 6: Vergleich bezüglich Performanz zwischen Blazor und Angular (Bourgarel 2020) | 31 |
| Tabelle 7: Konkurrenzvergleich anhand der funktionalen Anforderungen | 32 |
| Tabelle 8: Backlog | 40 |
| Tabelle 9: Testprotokoll | 46 |

9.5 Literaturverzeichnis

- Amsterdam Standard. *amsterdamstandard.com*. 2021. https://amsterdamstandard.com/en/post/the-benefits-of-working-with-an-agile-development-team (accessed 10 26, 2021).
- Athuraliya, Amanda. *creately.com.* 23 11 2021. https://creately.com/blog/diagrams/what-is-a-situation-analysis/ (accessed 12 23, 2021).
- Augsten, Stephen. *dev-insider.de*. 20 03 2020. https://www.dev-insider.de/was-ist-webassembly-a-912337/ (accessed 12 20, 2021).
- Babich, Nick. *xd.adobe.com*. 11 10 2019. https://xd.adobe.com/ideas/process/user-testing/everything-you-need-to-know-about-beta-testing/ (accessed 12 27, 2021).
- Bains, Callum. wargamer.com. 08 11 2021. https://www.wargamer.com/dnd/character-creator (accessed 12 29, 2021).
- Beresford, Phil. denofgeek.com. 25 01 2011. https://www.denofgeek.com/games/a-history-of-rpgs/ (accessed 12 29, 2021).
- Bourgarel, Rémi. *remibou.github.io.* 16 01 2020. https://remibou.github.io/Should-I-peek-Blazor/ (accessed 12 26, 2021).
- Chernyakov, Alexandr. *uptech.team.* 2021. https://www.uptech.team/blog/frontend-frameworks-for-web-product (accessed 10 26, 2021).
- Couriol, Bruno. *infoq.com.* 06 12 2019. https://www.infoq.com/news/2019/12/webassembly-w3c-recommendation/ (accessed 12 20, 2021).
- Ekwuno, Obinna. *blog.logrocket.com.* 16 12 2019. https://blog.logrocket.com/building-performant-web-applications-slow-networks/ (accessed 12 21, 2021).

- Foundry VTT. foundryvtt.com. 2021. https://foundryvtt.com/ (accessed 12 20, 2021).
- Freiler, Luke. *centercode.com*. 31 05 2019. https://www.centercode.com/blog/alpha-vs-beta-testing (accessed 12 22, 2021).
- Froyd, Nathan. *backs.mozilla.org*. 25 02 2020. https://hacks.mozilla.org/2020/02/securing-firefox-with-webassembly/ (accessed 12 20, 2021).
- GrabYourPitchforks. github.com. 27 09 2019. https://github.com/dotnet/runtime/issues/30969#issuecomment-535779492 (accessed 12 10, 2021).
- Hamilton, Thomas. *guru99.com.* 8 10 2021. https://www.guru99.com/software-testing-introduction-importance.html (accessed 11 12, 2021).
- Haslhofer, Johannes. *teilzeithelden.de.* 11 07 2020. https://www.teilzeithelden.de/2020/07/11/3-pen-and-paper-spielberichte-aus-der-corona-krise/ (accessed 10 20, 2021).
- Indeed Editorial Team. *indeed.com*. 17 09 2021. https://www.indeed.com/career-advice/career-development/user-interface (accessed 12 23, 2021).
- Johner, Christian. *johner-institut.de*. 05 08 2019. https://www.johner-institut.de/blog/tag/software-architektur/ (accessed 10 26, 2021).
- Jud, Martin, Jörg Hofstetter, Thomas Koller, and Ruedi Arnold. hsluzern.sharepoint.com. 02 05 2020. https://hsluzern.sharepoint.com/sites/i_projektschiene/SitePages/SoDa.aspx (accessed 10 12, 2021).
- Jung, Bernhard. *ulisses-ebooks.de.* 2021. https://www.ulisses-ebooks.de/product/212543/The-Dark-Aidalpha (accessed 12 20, 2021).
- Lai, Jonathan, and Andrew Chen. *future.a16z.com.* 01 11 2020. https://future.a16z.com/digital-future-tabletop-games/ (accessed 12 29, 2021).
- LePage, Pete. web.dev. 27 04 2020. https://web.dev/storage-for-the-web/ (accessed 11 01, 2021).
- Lin, Niki. *blog.amplexor.com.* 28 10 2020. https://blog.amplexor.com/de/ux-ui-und-konsistenz-wie-ein-gutes-designsystem-noch-besser-wird (accessed 10 26, 2021).
- Madan, Sumeet. *scrum.org.* 16 12 2019. https://www.scrum.org/resources/blog/done-understanding-definition-done (accessed 11 01, 2021).
- Malavasi, Alexandre. *medium.com*. 19 11 2020. https://medium.com/@alexandre.malavasi/top-10-nice-free-blazor-components-b42875e56b28 (accessed 10 26, 2021).
- MDN Web Docs. *developer.mozilla.org*. 2021. https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps/Offline_Service_workers (accessed 12 12, 2021).

- Microsoft. *docs.microsoft.com*. 2021. https://docs.microsoft.com/en-us/aspnet/core/blazor/hosting-models?view=aspnetcore-6.0 (accessed 12 20, 2021).
- Montag, Torsten. gruenderlexikon.de. 2021. https://www.gruenderlexikon.de/checkliste/informieren/marktanalyse/marktnischen-finden/ (accessed 12 23, 2021).
- Montelli, Chrissy. *businessinsider.com.* 28 05 2021. https://www.businessinsider.com/npc-meaning?r=US&IR=T (accessed 11 01, 2021).
- Morinigo, Laura. *medium.com.* 26 11 2019. https://medium.com/samsung-internet-dev/progressive-web-apps-making-app-like-experiences-in-the-browser-a15bd388963b (accessed 12 20, 2021).
- Morlion, Peter. *dzone.com.* 30 06 2018. https://dzone.com/articles/software-architecture-the-5-patterns-you-need-to-k (accessed 10 31, 2021).
- Moz. moz.com. 2021. https://moz.com/learn/seo/what-is-seo (accessed 12 26, 2021).
- Muñoz, Alvaro, and Oleksandr Mirosh. Friday the 13th JSON Attacks. PDF. HPE Software Security Research. July 2017.
- Oakes, Tiger, and Thomas Steiner. web.dev. 19 05 2021. https://web.dev/maskable-icon/ (accessed 12 26, 2021).
- Obermann, Lukas. *ulisses-ebooks.de*. 2021. https://www.ulisses-ebooks.de/product/220253/Optolith-Character-Generator (accessed 12 20, 2021).
- Osmani, Addy. *medium.com.* 15 08 2016. https://medium.com/dev-channel/offline-storage-for-progressive-web-apps-70d52695513c (accessed 11 01, 2021).
- Ponte, Dean Del. grails.org. 22 06 2018. https://grails.org/blog/2018-06-22.html (accessed 12 22, 2021).
- Pospelova, Polly. *deleteagency.com.* 14 11 2019. https://www.deleteagency.com/blog/how-to-get-a-100-percents-lighthouse-performance-score (accessed 12 26, 2021).
- Radigan, Dan. *atlassian.com*. 2021. https://www.atlassian.com/agile/project-management/estimation (accessed 12 27, 2021).
- Rupp, Chris. Requirements-Engineerung und -Management. 7. Auflage. Nürnberg: Carl Hanser Verlag München, 2021.
- Schönleben, Dominik. 24 08 2015. https://www.gq-magazin.de/auto-technik/article/das-schwarze-auge-sollte-sich-fur-seine-5-edition-ein-beispiel-deutschen-indie (accessed 12 29, 2021).
- Schwarz, Nela, and Saskia Grote. *meltwater.com.* 28 01 2020. https://www.meltwater.com/de/blog/mobile-first (accessed 10 24, 2021).
- Shafei, Shoresh. 23 10 2020. https://towardsdatascience.com/data-quality-for-everyday-analysis-d3aa1442c31 (accessed 12 27, 2021).

- Stegner, Ben. *makeuseof.com.* 04 11 2020. https://www.makeuseof.com/what-are-rpgs-role-playing-games/ (accessed 12 29, 2021).
- Theo, Charis. *charistheo.io.* 13 03 2021. https://www.charistheo.io/blog/2021/03/workbox-strategies-with-examples-and-use-cases/ (accessed 12 14, 2021).
- Toonen, Edwin. *yoast.com.* 06 05 2020. https://yoast.com/what-is-a-progressive-web-app-pwa/ (accessed 12 20, 2021).
- Trivedi, Jignesh. *c-sharpcorner.com*. 23 02 2020. https://www.c-sharpcorner.com/article/understand-dependency-injection-in-blazor/ (accessed 12 10, 2021).
- Tychsen, Anders, Michael Hitchens, Thea Brolund, and Manolya Kavakli. The Game Master. PDF. 2015.
- Ulisses Spiele GmbH. *ulisses-ebooks.de*. 2021. https://www.ulisses-ebooks.de/cc/7/scriptoriumaventuris (accessed 10 24, 2021).
- —. *ulisses-ebooks.de*. 2021. https://www.ulisses-ebooks.de/product/211558/DSA5-Deluxe-Heldendokumente-PDF-als-Download-kaufen?term=Delux (accessed 12 20, 2021).
- Vogel, Peter. *telerik.com.* 22 10 2020. https://www.telerik.com/blogs/is-blazor-safe-enterprise-bet (accessed 12 20, 2021).
- W3C. w3.org. 2021. https://www.w3.org/ (accessed 12 20, 2021).
- Wayner, Peter. *techbeacon.com.* 01 02 2015. https://techbeacon.com/app-dev-testing/top-5-software-architecture-patterns-how-make-right-choice (accessed 10 26, 2021).
- Ziemoński, Grzegorz. *dzone.com.* 13 02 2017. https://dzone.com/articles/layered-architecture-is-good (accessed 10 26, 2021).