



# **Flexible Bereitstellung von Arbeitsblättern für den sonderpädagogischen Unterricht**

Studienarbeit, Herbstsemester 2024

OST - Ostschweizer Fachhochschule  
Campus Rapperswil-Jona

**Datum:** 20. Dezember 2024

**Autoren:** Dario Berther  
Dejan Bogdanovic  
Fabrice Bosshard

**Betreuer:** Prof. Dr.-Ing. Frieder Loch

## Abstract

Lehrmittel für Schulen sind oft nicht leicht anpassbar, da den Schulen meistens keine bearbeitbaren Materialien zur Verfügung stehen. Diese Problematik betrifft vor allem Lehrkräfte mit heilpädagogischem Hintergrund, da sie Arbeitsblätter auf die Bedürfnisse ihrer Schüler mit Lernschwächen anpassen wollen. Standardisierte Materialien können Kinder mit Lernschwächen schnell überfordern oder unverständlich sein. Durch entsprechende Individualisierungen lässt sich der Lernprozess der Kinder extrem fördern. Diese Anpassungen umfassen nicht nur gestalterische Elemente wie Schriftgrößen, Farben oder Layouts, sondern auch Visualisierungen, um die Aufgaben verständlicher darzustellen. Die Anpassungen sind in der Praxis zeitintensiv und technisch umständlich umzusetzen.

Es soll eine innovative Lösung entwickelt werden, welche Lehrpersonen eine einfache Anpassung von Arbeitsblättern ermöglicht. Die Lösung benötigt ein benutzerfreundliches Webinterface, auf welchem die Lehrkräfte die Möglichkeit haben, Aufgaben flexibel zu gestalten und auf die spezifischen Bedürfnisse ihrer Schüler eingehen zu können.

In dieser Arbeit wurde ein Prototyp entwickelt, welcher die Spezifikationen aus der Anforderungsanalyse erfüllt und in Usability-Tests evaluiert wurde. Basierend auf den gewonnenen Erkenntnissen wurden Verbesserungsvorschläge und Erweiterungsmöglichkeiten formuliert, welche als Grundlage für eine zukünftige Weiterentwicklung dienen. Der Prototyp beweist, dass es möglich ist, Arbeitsblätter effizient und benutzerfreundlich zu individualisieren und damit die Lehrpraxis nachhaltig zu unterstützen.

## Management Summary

**Problemstellung** Lehrpersonen stehen vor der Herausforderung, Arbeitsblätter effizient an die Bedürfnisse der Kinder anzupassen. Viele physische Lehrmittel müssen zunächst eingescannt werden, was zu schlecht optimierten digitalen Dokumenten führt, die sich kaum weiter bearbeiten lassen. Bei digitalen Lehrmitteln wie z.B. PDFs, sind Anpassungen an Schriftgrösse oder dem Inhalt zwar möglich, jedoch andere Aspekte zeitaufwendig und technisch anspruchsvoll.

Hierzu zählen zum Beispiel, das Abdecken von Bildern, welche störend oder ablenkend sind, das Anpassen des Layouts der Arbeitsblätter, was oft mit ungenauen Abständen und ungleichmässigen Rändern einhergeht oder auch das Hinzufügen von Illustrationen oder weiteren Elementen, um das Verstehen und Lesen zu fördern. Diese manuelle Bearbeitung erfordert nicht nur viel Zeit, sondern birgt auch das Risiko von Fehlern mit sich. Als Lehrperson ist man dadurch stark im Handlungsspielraum eingeschränkt und verliert Zeit, welche für die eigentlich Unterrichtsvorbereitung genutzt werden kann. Es besteht ein dringender Bedarf nach einer effizienteren Lösung, die diese Prozesse automatisiert und vereinfacht.

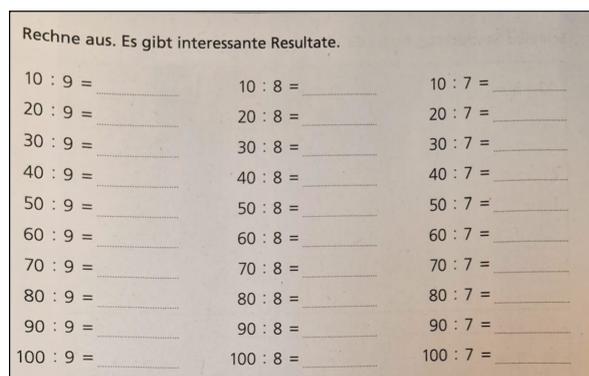


Abbildung 1: Überfülltes Arbeitsblatt  
Quelle: Arbeitsmaterial der Themenstellerin [34]

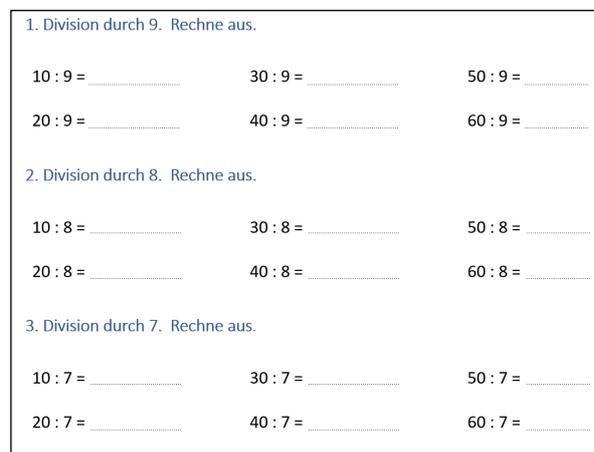


Abbildung 2: Manuell überarbeitetes Arbeitsblatt  
Quelle: Arbeitsmaterial der Themenstellerin [34]

Diese Problematik betrifft insbesondere Lehrpersonen aus der Schulischen Heilpädagogik. Sie unterrichten und unterstützen Kinder mit Lernschwächen. Zu den Lernschwächen gehören, unter anderem, Autismus, ADHS, Lese- und Rechtschreibschwächen, motorische Einschränkungen oder auch Deutsch als Zweitsprache. Für diese Kinder ist es wichtig, dass der Unterrichtsstoff auf ihre Bedürfnisse angepasst werden kann.

**Ziel der Arbeit** Ziel dieser Arbeit ist die Ausarbeitung und Umsetzung eines Konzepts, welches die Lehrkräfte bei der Erstellung von individuellen Arbeitsblättern unterstützt. Ein Prototyp soll zeigen, wie eine solche Applikation aussehen und funktionieren könnte. Der Fokus liegt auf folgenden Anpassungsmöglichkeiten:

- Dynamisches Anpassen des Layouts, z.B. Schriftgrössen, Abstände und Farben.
- Hinzufügen von visuellen Hilfsmitteln, wie Zahlenstrahlen oder Diagrammen, die den Lernprozess unterstützen.
- Erstellung von Vorlagen, die an die Bedürfnisse der Kinder angepasst werden können.
- Generierung von Arbeitsblättern im PDF-Format mit einem Layout, welches auf den Vorlagen basiert.

**Vorgehen** Das Vorgehen lässt sich in fünf Phasen aufteilen:

1. **Analyse:** In enger Zusammenarbeit mit der Themenstellerin wurden bestehende Arbeitsblätter analysiert, um ein Verständnis dafür zu entwickeln, was die Herausforderungen und Bedürfnisse sind.
2. **Konzeptentwicklung:** Um die Bedürfnisse abzudecken wurden mögliche Lösungsansätze recherchiert und ein technische Konzepte entworfen, wie diese Ansätze implementiert werden könnten.
3. **Ideenabgleich:** Für die Umsetzung der Arbeit wurde ein menschenzentrierter Designprozess genutzt, um die technische Umsetzungsmöglichkeiten mit der Vision der Themenstellerin abzugleichen.
4. **Implementierung:** Ein funktioneller Prototyp wurde entwickelt, der es ermöglicht, Arbeitsblätter dynamisch zu erstellen und zu individualisieren.
5. **Evaluierung:** Es wurden Usability-Tests durchgeführt, um die Benutzerfreundlichkeit und Effizienz der Applikation zu bewerten. Die gewonnenen Erkenntnisse dienten als Grundlage für zukünftige Optimierungen.

**Ergebnis** Der entwickelte Prototyp ermöglicht es Lehrpersonen, Mathematik-Arbeitsblätter flexibel zu gestalten. Die wichtigsten Funktionen sind:

- Erstellung und Bearbeitung von Aufgaben wie Addition, Subtraktion, Multiplikation und Division.
- Hinzufügen von visuellen Hilfsmitteln wie Zahlenstrahlen oder Tabellen, um das Verständnis zu fördern.
- Anpassung von Schriftgrößen, Abständen und Farben für eine einfache individuelle Gestaltung.
- Speichern von benutzerdefinierten Vorlagen zur Wiederverwendung für verschiedene Arbeitsblätter.

Die Umsetzung erfolgte mit Hilfe von Frameworks, welche aus HTML-Code PDFs generieren können. Ein Beispiel für ein generiertes PDF ist in Abbildung 3 dargestellt. Usability-Tests zeigten, dass Lehrpersonen den Prototyp nach kurzer Eingewöhnung bedienen können. Zusätzlich konnten aus den Usability-Test spezifische Verbesserungsvorschläge und Erweiterungsmöglichkeiten gewonnen werden.

**Fazit und Ausblick** Die Arbeit zeigt auf, dass es möglich ist Arbeitsblätter flexibel und benutzerfreundlich zu individualisieren. Der Prototyp bietet eine praktikable Lösung für die beschriebene Problematik und legt den Grundstein für zukünftige Erweiterungen. Im nächsten Schritt könnten weitere Aufgabentypen implementiert und die Digitalisierung bestehender physischer Lehrmittel erforscht werden. Zudem könnte die Integration zusätzlicher visueller Hilfsmittel ein grosses Potenzial, den Lernprozess weiter zu verbessern. Zudem könnte den Kindern die Möglichkeit geboten werden, die Arbeitsblätter digital innerhalb der Applikation zu lösen.

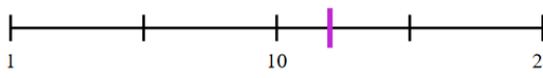
Name: \_\_\_\_\_  
 Klasse: \_\_\_\_\_

## Mathematik Übung 1

**1. Rechne die Aufgaben aus und schreibe die Resultate auf die Linie**

**1.1. Aufgabe**

$4 + 8 = \underline{\quad}$

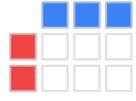


**1.2. Aufgabe**

$7 + 7 = \underline{\quad}$        $8 + 3 = \underline{\quad}$        $9 + 4 = \underline{\quad}$        $5 + 4 = \underline{\quad}$

**1.3. Aufgabe**

$3 * 2 = \underline{\quad}$



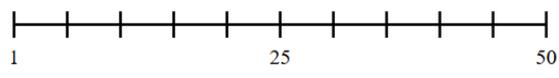
**1.4. Aufgabe**

$2 * 4 = \underline{\quad}$        $3 * 4 = \underline{\quad}$        $3 * 3 = \underline{\quad}$        $2 * 3 = \underline{\quad}$

**2. Zeichne das Ergebnis der Rechenaufgaben in die Darstellung**

**2.1. Aufgabe**

$18 + 5 \rightarrow$



**2.2. Aufgabe**

$3 + 6 \rightarrow$

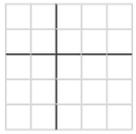


Abbildung 3: Generiertes PDF  
 Quelle: Aus Prototyp [28]

## Inhaltsverzeichnis

<b>1 Einführung</b>	<b>7</b>
1.1 Ausgangslage . . . . .	7
1.2 Ziel der Arbeit . . . . .	9
1.3 Vorgehen . . . . .	10
<b>2 Recherche</b>	<b>11</b>
2.1 Untersuchung technischer Ansätze . . . . .	11
2.2 Software- / Marktanalyse . . . . .	16
2.3 Pädagogische Strategien und soziologische Einflüsse . . . . .	18
<b>3 Evaluierung der Umsetzungstrategie</b>	<b>20</b>
<b>4 Umsetzung des User-Centered Design</b>	<b>22</b>
4.1 Initiale Mockups . . . . .	22
4.2 Ausarbeiten der Anforderungen . . . . .	32
4.3 Zweiter Austausch mit der Themenstellerin . . . . .	35
4.4 Usability Test . . . . .	36
<b>5 Anforderungen</b>	<b>39</b>
5.1 Funktionale Anforderungen . . . . .	39
5.2 Nicht-funktionale Anforderungen . . . . .	44
<b>6 Technologien</b>	<b>49</b>
6.1 Blazor . . . . .	49
6.2 MudBlazor . . . . .	49
6.3 ASP.NET Core Web API . . . . .	49
6.4 Entity Framework Core . . . . .	49
6.5 Andere Frameworks & Packages . . . . .	50
<b>7 Konzept und Architektur</b>	<b>52</b>
7.1 C4 Modell . . . . .	52
7.2 Infrastruktur . . . . .	68
<b>8 Technische Umsetzung</b>	<b>71</b>
8.1 Blazor Konzepte . . . . .	71
8.2 Swagger UI . . . . .	73
8.3 Generierung von Aufgaben . . . . .	74
8.4 PDF Generieren . . . . .	77
8.5 Authentifizierung . . . . .	80
<b>9 Resultate und Ausblick</b>	<b>82</b>
9.1 Resultate . . . . .	82
9.2 Usability Test - Testergebnisse . . . . .	91
9.3 Ausblick . . . . .	94
<b>10 Retrospektive</b>	<b>97</b>
10.1 Reflexion . . . . .	97
10.2 Fazit . . . . .	98
<b>Literaturverzeichnis</b>	<b>99</b>

<b>Glossar</b>	<b>100</b>
<b>A Testprotokoll</b>	<b>110</b>
A.1 Manuelle Tests . . . . .	110
<b>B Mockups</b>	<b>112</b>

# 1 Einführung

Zu Beginn der Arbeit wurde die Ausgangslage in einem Kick-off-Meeting mit der Themenstellerin besprochen. Ziel war es, besser zu verstehen wie die bestehende Problematik sowie ihre Vision für eine mögliche Lösung aussehen. Im Rahmen dieser Analyse stellte die Themenstellerin verschiedene Mathematik-Arbeitsblätter zur Verfügung, welche bereits angepasst wurden oder bei denen Anpassungen erforderlich wären.

## 1.1 Ausgangslage

Die Arbeitsblätter wurden analysiert, um die Anforderungen und Herausforderungen der Individualisierung besser zu verstehen. Ziel der Analyse war es, Ansatzpunkte für die Entwicklung einer Lösung herauszufinden, welche den Arbeitsaufwand reduzieren und die Anpassungsmöglichkeiten verbessern. Folgende Problemstellungen wurden identifiziert:

**Abdecken von störenden Bildern** In diesem Beispiel handelt es sich um ein Bild, welches in keinem direkten Zusammenhang mit den Mathematik-Aufgaben steht und für Kinder ablenkend wirken kann. Zwar ist eine spielerische Aufgabe zur Auflockerung der Konzentration durchaus hilfreich, jedoch sollte diese klar abgegrenzt sein oder gar nicht auf dem selben Arbeitsblatt wie die eigentlichen Aufgaben platziert werden.

Zur Verschleierung solcher Bilder, werden die betroffenen Stellen oft mit Rechtecken überdeckt. Diese Lösung ist unbefriedigend, da das überdeckende Rechteck vor allem beim Ausdruck auffällt, etwa durch Farbunterschiede oder sichtbare Konturen. Solche visuellen Ungleichheiten können weiterhin ablenken und beeinträchtigen die Konzentration der Schülerinnen und Schüler.

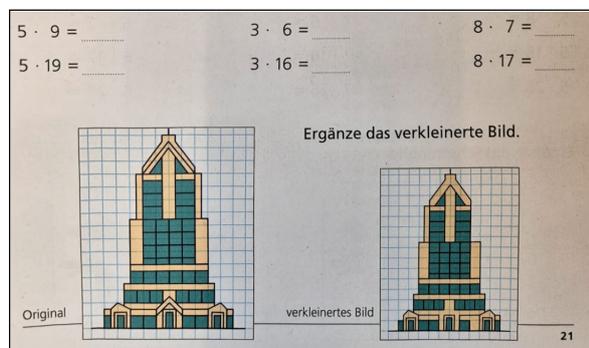


Abbildung 4: Ablenkendes Bild

Quelle: Arbeitsmaterial der Themenstellerin [34]

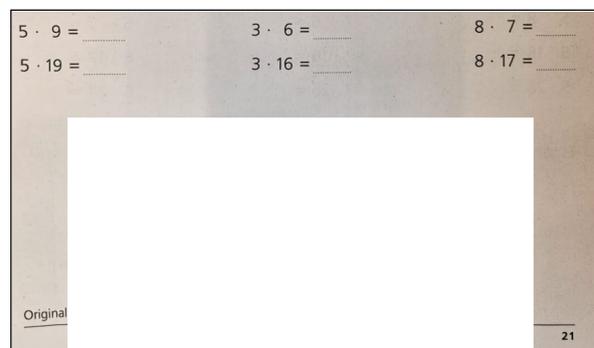


Abbildung 5: Abgeklebtes Bild

Quelle: Arbeitsmaterial der Themenstellerin [34]

Entweder akzeptiert man die Unschönheiten des Arbeitsblattes, oder man investiert viel Zeit, um das Layout des Arbeitsblattes selbst nachzubilden. Dabei geht wertvolle Zeit verloren, die für andere Aufgaben in der Unterrichtsvorbereitung genutzt werden könnte.

## Überfüllte Arbeitsblätter

Das nachfolgende Bild zeigt einen Ausschnitt aus einem Arbeitsblatt, welches durch die Vielzahl an Aufgaben und fehlenden Abständen zwischen den Zeilen überladen wirkt. Diese Gestaltung führt dazu, dass Kinder leichter Fehler machen, beispielsweise indem sie die Lösungen in der falschen Zeile eintragen. Besonders für Kinder mit Legasthenie stellt ein solches Layout eine zusätzliche Herausforderung dar.

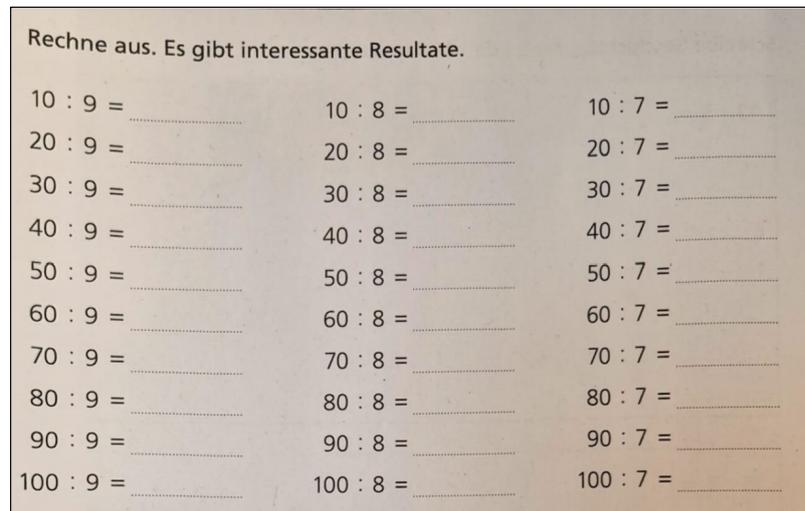


Abbildung 6: Überfülltes Aufgabenblatt

Quelle: *Arbeitsmaterial der Themenstellerin [34]*

Die in der nachfolgenden Abbildung ersichtlichen Änderungen sind im Vergleich klar strukturiert und mit ausreichenden Abständen zwischen den Aufgaben versehen, was die Lesbarkeit verbessert und kognitive Überforderung reduziert. Dies hilft insbesondere Kindern mit Legasthenie[20] oder Aufmerksamkeitsproblemen[1], sich besser zu orientieren und Fehler zu vermeiden.

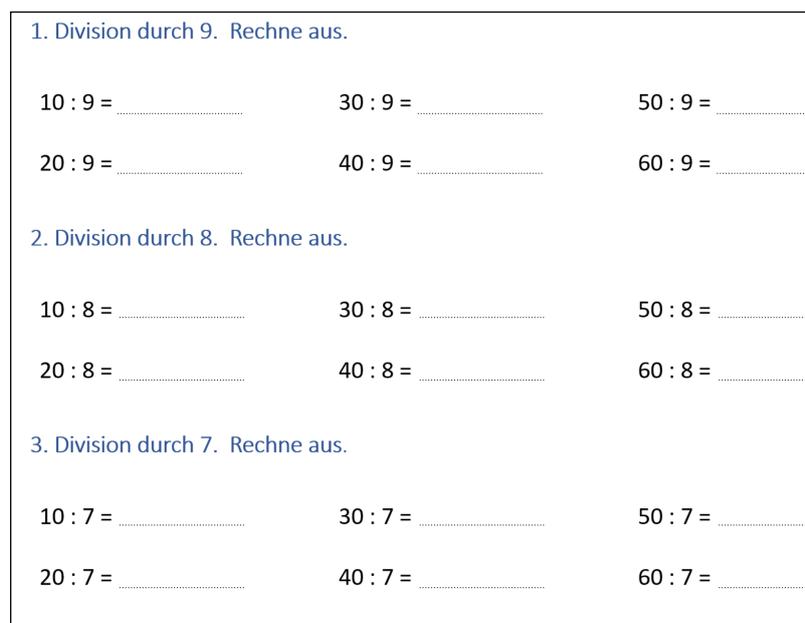


Abbildung 7: Überarbeitetes Layout des Aufgabenblattes

Quelle: *Eigene Erstellung [13]*

## Einfügen von Elementen

Ein weiteres Problem ist, dass gewisse Aufgaben aus Lehrmitteln nicht die nötigen stützenden Hilfestellungen enthalten. Die Themenstellerin teilte mit, dass Kinder mit Lernschwächen durch solch einfachen Anpassungen stark profitieren können. Im nachfolgenden Bild wurde rot umrandet, was von ihr angepasst wurde. Das Hinzufügen eines Striches zur Kennzeichnung des Lösungsortes sowie der ergänzten Stellen auf dem Zahlenstrahl sind einfache, jedoch zeitaufwendige Änderungen, die von Hand vorgenommen werden müssen.

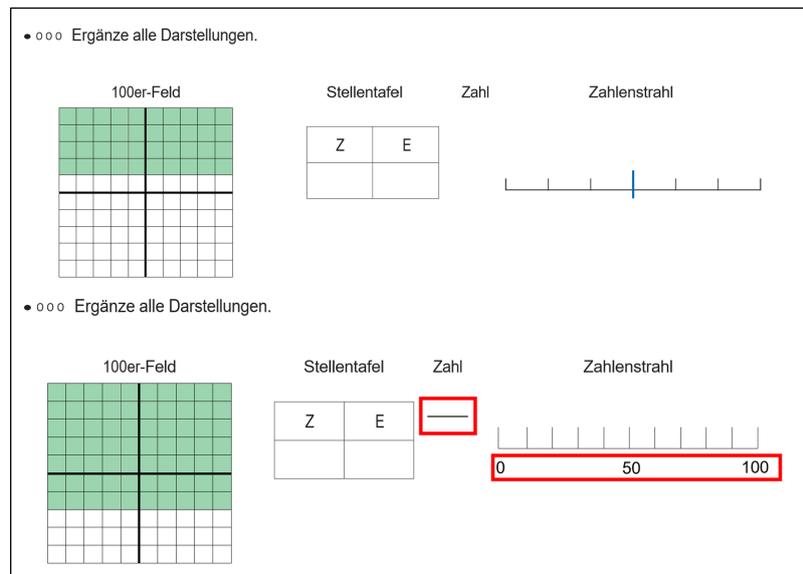


Abbildung 8: Hilfestellungen manuell hinzufügen  
 Quelle: *Arbeitsmaterial der Themenstellerin [34]*

Das Beispiel zeigt auf, wie wichtig visuelle Hilfsmittel bei der Erstellung von Arbeitsblättern sein können. Solche visuellen Stützen könnten während der Erstellung der Aufgabenblätter aktiviert und später automatisch generiert werden. Somit wären keine manuellen Schritte zur Gestaltung mehr nötig und gleichzeitig könnte das Verständnis der Schülerinnen und Schüler gefördert werden. Beispiele hierfür sind Zahlenstrahlen für Additions- und Subtraktionsaufgaben, Kuchendiagramme zur Veranschaulichung von Brüchen oder Tabellen für Multiplikationsaufgaben.

## 1.2 Ziel der Arbeit

Bei der Analyse der Ausgangslage haben sich zwei Ansätze zur Lösung des Problems herauskristallisiert.

Der erste Ansatz dreht sich um das Einlesen bestehender physischer Lehrmittel. Hierbei soll es möglich sein, gescannte Arbeitsblätter in die Applikation einzulesen und direkt digital anzupassen. Dies erfordert jedoch eine detaillierte Analyse der Bilder um einzelne Elemente, wie Aufgaben, Text, und Illustrationen erkennen und später weiterverarbeiten zu können.

Der zweite Ansatz konzentriert sich auf das dynamische Erstellen von digitalen Arbeitsblättern. Dabei geht es nicht nur darum, bestehende Arbeitsblätter einfach und schnell anpassen zu können, sondern auch die Möglichkeit zu bieten, Arbeitsblätter und deren Layouts von Grund auf neu zu gestalten. Hierfür muss ein generischer Weg gefunden werden, um ein Arbeitsblatt technisch zu beschreiben und dessen Layout, Inhalt sowie visuelle Elemente dynamisch zu generieren und zu speichern. Mit diesem Ansatz wird ein Fundament geschaffen, welches später

um komplexere Funktionalitäten wie die Digitalisierung bestehender Materialien erweitert werden könnte.

Die Umsetzung dieser Arbeit beschränkt sich auf die Entwicklung eines Konzepts mit einer prototypischen Implementierung. Ziel des Konzeptes ist es technisch aufzuzeigen, wie Arbeitsblätter in digitaler Form erstellt sowie angepasst werden können um dabei den Lehrpersonen viel Zeit einzusparen. Das zentrale Ergebnis des Prototyps soll die Möglichkeit sein, die erstellten Arbeitsblätter in PDF-Format zu generieren. Der Schwerpunkt liegt zunächst auf dem Fach Mathematik, da hier klare Strukturen und Aufgabenformate bestehen, welche sich gut für eine prototypische Umsetzung eignen.

### 1.3 Vorgehen

Um die gesetzten Ziele zu erreichen, wurde ein Vorgehen gewählt, welches sich aus mehreren Phasen zusammensetzt:

**Technische Recherche und Entscheidungsfindung** Zu Beginn der Arbeit wird eine technische Recherche durchgeführt (siehe Kapitel 2.1), um geeignete Technologien und Lösungsansätze zu identifizieren. Hierbei sollen sowohl Ansätze zur Verarbeitung bestehender Lehrmittel als auch zur dynamischen Erstellung von Arbeitsblättern untersucht werden.

**Anforderungen und Design** In enger Zusammenarbeit mit der Themenstellerin sollen die Anforderungen an die Lösung präzisiert werden. Darauf aufbauend wird ein benutzerzentriertes Design entwickelt, das die wichtigsten Funktionen des Systems abdeckt. Dabei sollen geeignete Technologien evaluiert werden um eine Webapplikation zu entwickeln, welche den Ansprüchen entspricht.

**Iterative Entwicklung des Prototyps** Die prototypische Umsetzung erfolgt iterativ, um frühzeitig Feedback zu sammeln und mögliche Anpassungen zu integrieren.

**Validierung durch Usability-Tests** Um die Benutzerfreundlichkeit und Effizienz des Prototyps sicherzustellen, wird der Prototyp in Usability-Tests mit mehreren Lehrpersonen getestet. Die Tests zielten darauf ab, die Intuitivität der Benutzeroberfläche zu überprüfen und mögliche Optimierungen zu finden. Die gewonnenen Erkenntnisse werden dokumentiert und fließen direkt in die Verbesserungsvorschläge ein.

## 2 Recherche

In diesem Kapitel werden die Recherchearbeit und die daraus gewonnenen Erkenntnisse beschrieben. Es wurde eine Kombination aus technischer und sozialpädagogischer Recherche durchgeführt, um ein Basiswissen aufzubauen, welches zur Unterstützung und Wegweisung dieser Arbeit beiträgt. Zudem wurden bestehende Applikationen analysiert, wobei deren Stärken und Schwächen analysiert und bewertet wurden.

### 2.1 Untersuchung technischer Ansätze

Zur Vertiefung der beiden im Kapitel 1.2 beschriebenen Ansätze wurde je ein entsprechendes technologisches Konzept untersucht. Das Ziel dieser Untersuchung war es, bestehende technische Möglichkeiten zu untersuchen um besser entscheiden zu können, welcher Ansatz umsetzbar ist. Im nachfolgenden Kapitel 3 wird das gewonnene Wissen reflektiert und eine Entscheidung für die weitere Umsetzung getroffen.

#### 2.1.1 Optical Character Recognition (OCR)

Der erste Ansatz fokussiert sich auf die selbstständige Digitalisierung physischer Lehrmittel, welche eine grosse Herausforderung für Lehrpersonen darstellt. Lehrmittel können zwar mithilfe eines Scanners in ein PDF umgewandelt werden, die resultierenden PDFs sind jedoch meist nur schwer editierbar. Um das eingescannte Dokument einfacher editieren zu können, bieten sich andere bearbeitbare Dateiformate an (z.B. Microsoft Word). Eine Konvertierung in solche Formate kann zwar durchgeführt werden, ist jedoch oft mit fehlerhaften Formatierungen oder fehlenden Abschnitten verbunden.

OCR[27] ist eine Technologie zur automatischen Erkennung von Texten aus Bildern oder gescannten Dokumenten. Sie ermöglicht es, gedruckten oder handgeschriebenen Text in eine digitale, editierbare Form umzuwandeln. Der Einsatz von dieser Technologie könnte somit massgeblich zur Lösung der Scan-Problematik beitragen, indem gescannte Arbeitsblätter direkt als bearbeitbare Vorlagen genutzt werden könnten.

**Funktionsweise** OCR-Systeme arbeiten typischerweise in mehreren Schritten, um die Genauigkeit der Texterkennung zu maximieren:

1. **Bildvorverarbeitung:** Eingescannte Bilder werden zunächst optimiert, um die Erkennungsgenauigkeit zu erhöhen. Dieser Schritt umfasst:

- *Rauschreduktion:* Entfernen von Störungen wie Staubpartikel oder Flecken im Hintergrund, um die Genauigkeit der Texterkennung zu verbessern.
- *Kontrastanpassung:* Optimierung der Lesbarkeit durch Anpassung von Helligkeit und Kontrast. Ein höherer Kontrast erleichtert die Trennung von Text und Hintergrund.
- *Schwarz-Weiss-Konvertierung:* Konvertierung des Bildes in Schwarz-Weiss, um Buchstaben und Symbole klarer hervorzuheben.

2. **Texterkennung:** Die Texterkennung selbst erfolgt durch Mustererkennung oder maschinelles Lernen. Dabei werden Zeichenmuster im Bild mit einer Datenbank bekannter Schriftzeichen abgeglichen. Fortschrittliche Algorithmen analysieren zusätzlich:
  - *Layout-Analyse:* Erkennung von Zeilen, Absätzen und Strukturen wie Tabellen oder Listen.
  - *Form- und Linienerkennung:* Identifikation grafischer Elemente, die nicht zu reinem Text gehören (z. B. Linien, Rahmen).
3. **Post-Processing:** Nach der Texterkennung wird der Text durch Rechtschreibprüfung oder Korrekturalgorithmen verbessert.

**Anwendungsgebiete** OCR findet in verschiedenen Bereichen Anwendung:

- **Digitale Archivierung und Dokumentenmanagement:** Automatisches Erfassen und Speichern von Textinhalten zur langfristigen, durchsuchbaren Archivierung.
- **Automatisierung von Datenverarbeitung:** Extraktion von Text aus z.B. Formularen, Rechnungen oder Verträgen zur digitalen Verarbeitung.
- **Texterkennung für Blinde und Sehbehinderte:** Unterstützung durch Screenreader und Sprachausgabe, um digitale Inhalte zugänglich zu machen.
- **Post- und Versandetikettenverarbeitung:** Automatische Erkennung von Adressen und Barcodes zur Optimierung von Logistikprozessen.
- **Nummernschilderkennung:** Einsatz in der Verkehrsüberwachung und automatisierten Mautsystemen.
- **Erkennung von Büchern und Manuskripten:** Digitalisieren und Archivieren von gedruckten oder historischen Texten.
- **Verarbeitung handgeschriebener Texte:** Umwandeln von handschriftlichen Dokumenten in digitale Formate.
- **Bearbeitung von PDF-Dokumenten:** Konvertierung von gescannten PDFs in bearbeitbare Textdokumente.

### Herausforderungen

- **Qualität der Eingangsbilder:** Schlechte Bildqualität, hohes Hintergrundrauschen oder verzerrte Scans verringern die Genauigkeit der Texterkennung.
- **Handgeschriebener Text:** Handschriftliche Texte sind aufgrund ihrer Variabilität schwer zu erkennen, obwohl moderne Systeme wie Google Cloud Vision und spezialisierte Tools wie MathPix bedeutende Fortschritte gemacht haben. Auch mathematische Formeln können schnell fehlinterpretiert werden.
- **Sprachuntersützung:** Einige Sprachen oder Schriftsysteme, wie arabische oder chinesische Schriftzeichen, sind aufgrund ihrer Komplexität schwieriger zu erkennen.
- **Layout-Erkennung:** OCR-Systeme haben Schwierigkeiten, komplexe Layouts zu interpretieren, z.B. Tabellen, mehrspaltige Texte oder eingebettete Bilder.

**Fazit zur Relevanz von OCR** OCR bietet die Möglichkeit, physische Lehrmittel zu digitalisieren und bearbeitbar zu machen, was insbesondere bei bestehenden Arbeitsblättern von Vorteil wäre. Die Technologie könnte es Lehrpersonen ermöglichen Texte und Layouts anzupassen, ohne die Arbeitsblätter von Hand neu erstellen zu müssen. Allerdings erfordert OCR eine gute Bildqualität und stösst bei komplexen Layouts, handschriftlichen Notizen oder mathematischen Formeln schnell an ihre Grenzen. Die Nachbearbeitung kann somit einen erheblichen Mehraufwand mit sich bringen, was den angestrebten Zeitgewinn für Lehrpersonen reduziert.

### 2.1.2 PDF-Generator

Der zweite Ansatz konzentriert sich auf die dynamische Erstellung von Arbeitsblättern im PDF-Format. Aufgaben können dabei nicht nur inhaltlich, sondern auch strukturell flexibel gestaltet werden, um den spezifischen Bedürfnissen der Schülerinnen und Schüler gerecht zu werden.

Zur Darstellung der Arbeitsblätter in einer Webapplikation würden HTML und CSS verwendet werden. Wenn nun das PDF auf Basis dieser dynamischen Web-Komponenten erstellt werden könnten, wäre das ein grosser Vorteil, da die Komponenten nicht doppelt erstellt werden müssten. Das direkte Rendern von HTML-Inhalten zu PDFs würde eine schnelle und effiziente Erstellung von Arbeitsblättern ermöglichen und die technische Komplexität reduzieren.

Es gibt auf dem Markt zahlreiche Tools, die für die Erstellung von PDFs genutzt werden können. Folgende Tools wurden evaluiert:

**IronPDF** IronPDF[5] ist ein sehr benutzerfreundliches Tool, das sich besonders durch seine einfache Handhabung auszeichnet. Der grösste Vorteil dieses Tools liegt in der direkten Umwandlung von HTML in PDF-Dokumente. Die Dokumentation ist ausführlich und bietet viele praktische Beispiele. Allerdings ist IronPDF kostenpflichtig, was gerade für Schulprojekte ein Nachteil ist. Auch sind die Lizenzen nicht kostengünstig. Die günstigste Lizenz kostet 749 Dollar pro Jahr. Somit fällt IronPDF im Rahmen dieser Arbeit raus.

**Which perpetual license fits your project needs?**  
Coverage based on number of [developers](#), [locations](#), and [projects](#).

IRONPDF	IRONPDF	IRONPDF	IRONPDF
<p><b>Lite</b> <b>\$749 USD</b></p> <ul style="list-style-type: none"> <li>✓ 1 developer</li> <li>✓ 1 location</li> <li>✓ 1 project</li> </ul> <p>♥ Email support</p>	<p><b>Plus</b> <b>\$1,499 USD</b></p> <ul style="list-style-type: none"> <li>✓ 3 developers</li> <li>✓ 3 locations</li> <li>✓ 3 projects</li> </ul> <p>♥ Email support (48h SLA) ♥ 24/5 Live Chat support</p>	<p><b>Professional</b> <b>\$2,999 USD</b></p> <p><b>MOST POPULAR</b></p> <ul style="list-style-type: none"> <li>✓ 10 developers</li> <li>✓ 10 locations</li> <li>✓ 10 projects</li> </ul> <p>♥ Email support (24h SLA) ♥ 24/5 Live Chat support ♥ Screen-sharing support</p>	<p><b>Unlimited</b> <b>\$5,999 USD</b></p> <ul style="list-style-type: none"> <li>✓ Unlimited developers</li> <li>✓ Unlimited locations</li> <li>✓ Unlimited projects</li> </ul> <p>♥ Email support (24h SLA) ♥ 24/5 Live Chat support ♥ Screen-sharing support</p> <p>For best coverage consider <b>Enterprise</b></p>

Abbildung 9: Preismodelle von IronPDF

Quelle: <https://ironpdf.com/licensing/>

**iText7** iText7[32] ist ein leistungsstarkes und flexibles Werkzeug für die PDF-Generierung. Es bietet eine Vielzahl von Funktionen, wie zum Beispiel die Erstellung von Formularen oder die Einbindung digitaler Unterschriften. Ein Vorteil ist die Verfügbarkeit einer kostenlosen Community-Version, die grundlegende Funktionen abdeckt. Allerdings erfordert die Arbeit mit iText7 Programmierkenntnisse und Zeit, da die Lernkurve steil ist und die Erstellung von PDFs oft sehr detaillierte Anpassungen erfordert. Besonders in professionellen Projekten, die spezifische und komplexe Anforderungen an die PDF-Erstellung haben, spielt iText7 seine Stärken aus.

**QuestPDF** QuestPDF[24] hingegen zeichnet sich durch seine moderne und unkomplizierte Programmierung aus. Das Tool ist vollständig kostenlos, Open Source und bietet gut strukturierten Code, welcher das Erlernen erleichtert.

Der Unterschied zu den anderen Tools besteht darin, dass QuestPDF keine direkte Umwandlung von HTML in PDF ermöglicht. Stattdessen basiert es auf einer FluentAPI, bei der PDFs direkt aus dem C#-Code generiert werden. Zur Erstellung werden Methoden schrittweise miteinander verkettet (siehe Abbildung 10). Diese Vorgehensweise bietet zwar keine Wiederverwendung von bestehendem HTML, ermöglicht jedoch eine präzise Steuerung der PDF-Inhalte.

Im Vergleich zu iText7 bietet es zwar weniger Funktionen und ist auch noch relativ neu auf dem Markt, aber für die meisten Standard-Anforderungen völlig ausreichend.

```
using QuestPDF.Fluent;
using QuestPDF.Helpers;
using QuestPDF.Infrastructure;

Document.Create(container =>
{
    container.Page(page =>
    {
        page.Margin(50);
        page.Size(PageSizes.A4);
        page.PageColor(Colors.White);
        page.DefaultTextStyle(x => x.FontSize(16));

        page.Header()
            .AlignCenter()
            .Text("Invoice #: 2023-77")
            .SemiBold().FontSize(24).FontColor(Colors.Grey.Darken4);

        page.Content()
            .Table(table =>
            {
                table.ColumnsDefinition(columns =>
                {
                    columns.ConstantColumn(20);
                    columns.RelativeColumn();
                    columns.RelativeColumn();
                });

                table.Header(header =>
                {
                    header.Cell().Text("#");
                    header.Cell().Text("Product");
                    header.Cell().AlignRight().Text("Price");
                });

                foreach (var lineItem in lineItems)
                {
                    table.Cell().Text(lineItem.Index.ToString());
                    table.Cell().Text(lineItem.Name);
                    table.Cell().Text($"{lineItem.Price}");
                }
            });
    });
})
.GeneratePdf("invoice.pdf");
```

Abbildung 10: Beispiel einer FluentAPI in QuestPDF  
Quelle: Erstellt mit ClaudeAI

## 2.2 Software- / Marktanalyse

Als Inspiration für den Prototypen wurden zwei bestehende Plattformen zur Erstellung von Arbeitsblättern analysiert: Worksheet Crafter und Canva. Die Untersuchung konzentrierte sich auf deren Funktionsumfang, technische Einschränkungen sowie typische Herausforderungen, welche Lehrpersonen bei der Nutzung dieser Tools erleben könnten.

### 2.2.1 Worksheet Crafter

Worksheet Crafter[37] bietet umfangreiche Möglichkeiten zur Gestaltung von Arbeitsblättern. Die Software bietet einen umfangreichen Baukasten mit zahlreichen Vorlagen, Illustrationen und Werkzeugen zur Erstellung von Aufgaben, die sich visuell an die Bedürfnisse der Schülerinnen und Schüler anpassen lassen.

Allerdings hat das Tool mehrere Einschränkungen:

- **Kostenpflichtige Lizenz:** Die Nutzung erfordert den Erwerb einer kostenpflichtigen Lizenz.
- **Plattformabhängigkeit:** Worksheet Crafter ist nur für Windows und macOS verfügbar. Eine Webversion oder Unterstützung für Tablets fehlt.
- **Komplexität des Layouts:** Trotz des grossen Funktionsumfangs benötigt die Anpassung des Layouts viel Zeit. Auf neue Nutzer kann der Funktionsumfang überwältigend wirken.

Nichtsdestotrotz bietet Worksheet Crafter eine gute Inspirationsquelle, wie man verschiedene Aufgaben visuell gestalten kann, um sie individuell an die Bedürfnisse der Kinder anzupassen.

#### Beispiele für Aufgaben mit Illustrationen:

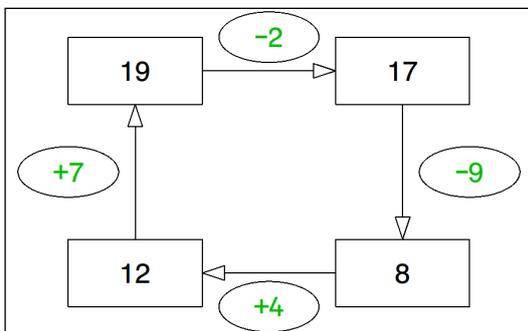


Abbildung 11: Illustrationsbeispiel von Worksheet Crafter

Quelle: Erstellt mit Worksheet Crafter [37]

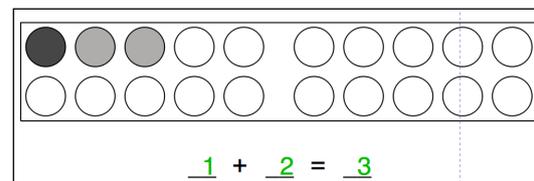


Abbildung 12: Illustrationsbeispiel von Worksheet Crafter

Quelle: Erstellt mit Worksheet Crafter [37]

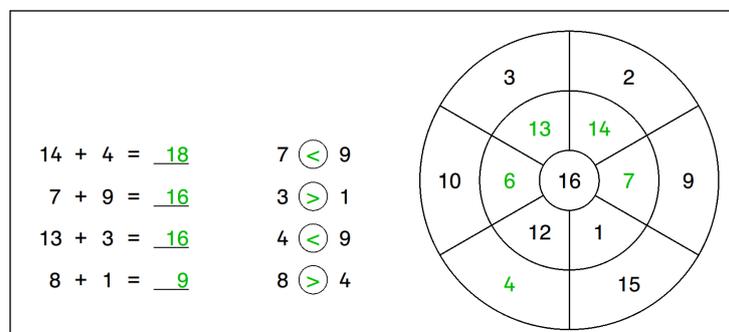


Abbildung 13: Illustrationsbeispiel von Worksheet Crafter

Quelle: Erstellt mit Worksheet Crafter [37]

### 2.2.2 Canva

Eine weitere Möglichkeit zur Erstellung von Arbeitsblättern und anderen Materialien ist die Plattform Canva[6]. Canva bietet eine kostenfreie Version sowie kostenpflichtige Pro- und Enterprise-Lizenzen, die den Funktionsumfang erweitern. Die Plattform ermöglicht es Inhalte per Drag-and-Drop zu gestalten, was die Erstellung von Arbeitsblättern erleichtert.

Der Funktionsumfang beinhaltet eine grosse Auswahl an Vorlagen, Bildern, Schriftarten und Illustrationen. Die Vorlagen für Mathematik oder andere Fachbereiche bieten Lehrpersonen eine gute Grundlage.

Trotz der Vorteile stösst Canva bei spezifischen Anforderungen an seine Grenzen:

- **Zeitaufwand für das Layout:** Das manuelle Gestalten und Anpassen des Layouts erfordert weiterhin viel Zeit, insbesondere bei komplexeren Arbeitsblättern.
- **Statische Inhalte:** Die integrierten Illustrationen, wie z.B. Darstellungen für Brüche oder Zahlenstrahlen, sind statisch. Das heisst sie sind nicht dynamisch veränderbar, da die Werte nicht an die Illustrationen gekoppelt sind. Die Illustrationen müssen demnach manuell angepasst werden, was umständlich ist.

Canva bietet zwar eine benutzerfreundliche Handhabung und verschiedene Gestaltungsmöglichkeiten, jedoch fehlt die notwendige Automatisierung bei der Erstellung von Illustrationen. Für den Einsatz in der Erstellung von Arbeitsblättern für Kinder mit speziellen Bedürfnissen ist dieser Aspekt ein wesentlicher Nachteil.

#### Beispiele Vorlagen:

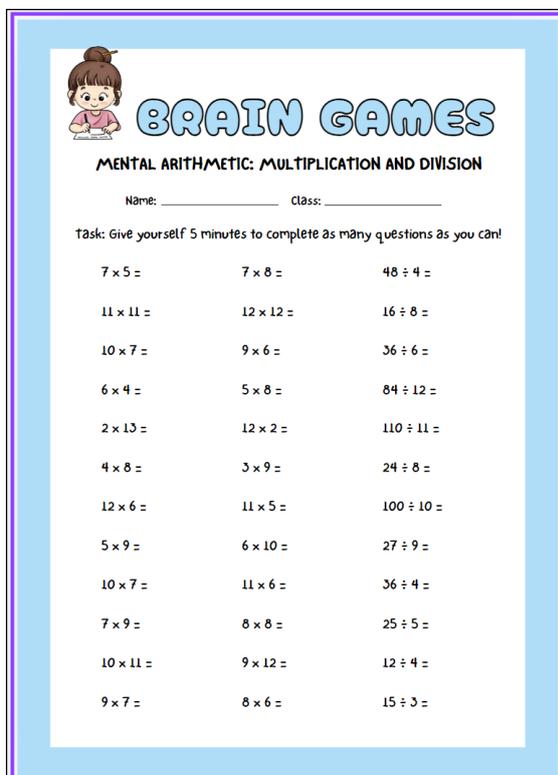


Abbildung 14: Vorlagenbeispiel von Canva  
 Quelle: Erstellt mit Canva [6]

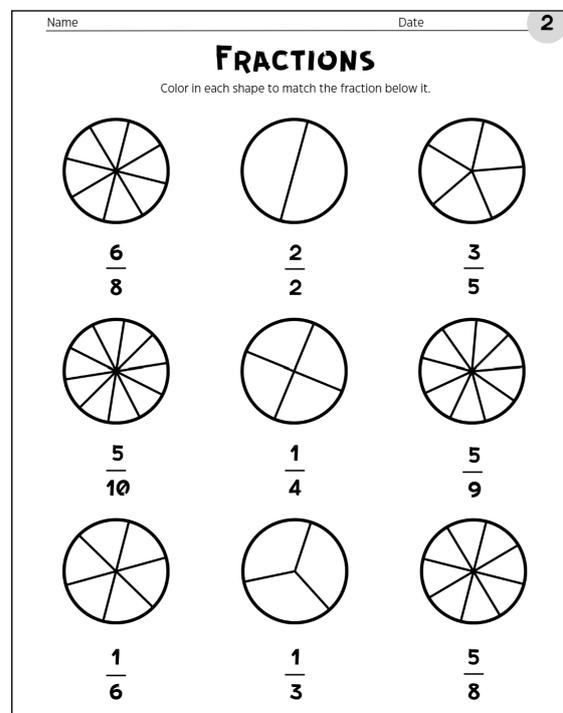


Abbildung 15: Illustrationsbeispiel von Canva (Nicht dynamisch veränderbar)  
 Quelle: Erstellt mit Canva [6]

## 2.3 Pädagogische Strategien und soziologische Einflüsse

Ein wichtiger Baustein dieser Arbeit ist die Eingrenzung der Funktionalitäten des Prototyps. Da die Themenstellerin mit Kindern mit Lernschwächen arbeitet und Arbeitsblätter individuell an deren Bedürfnisse anpasst, wurde eine sozialpädagogische Recherche durchgeführt. Ziel war es, Kriterien zu identifizieren, welche die Unterstützung dieser Kinder gewährleisten können. Hierbei wurden verschiedene Lernschwächen analysiert und deren Anforderungen an die Gestaltung von Arbeitsblättern dokumentiert.

**Kinder mit Legasthenie** Kinder mit Legasthenie[20] leiden an einer Lese-Rechtschreib-Schwäche. Ein Informationsblatt von worksheet crafter[38] empfiehlt auf folgende Kriterien zu achten, um diesen Kindern zu helfen:

- **Grosse Schriftgrösse:** Mindestens Schriftgrösse 14.
- **Grosser Zeilenabstand:** Mindestens anderthalbfacher Zeilenabstand.
- **Einfache Schriftart:** Verwendung von serifenlosen Schriftarten oder spezielle Schriftarten für LSRLRS, wie OpenDyslexic (Allerdings sind die wissenschaftlichen Studien[38] zufolge nicht effektiver als Arial).
- **Hervorhebungen:** Wichtige Wörter oder Sätze durch Unterstreichung oder Fettdruck hervorheben.

**Kinder mit Dyskalkulie** Kinder mit Dyskalkulie[11] leiden an einer Rechenschwäche. Nachfolgende Kriterien können eine Unterstützung sein.[29]

- **Visuelle Hilfsmittel:** Verwendung von visuellen Hilfsmitteln wie Bilder, Diagramme oder weitere Illustrationen, um mathematische Konzepte zu erklären.
- **Schritt für Schritt-Erklärungen:** Komplexe Rechenaufgaben in kleine, überschaubare Schritte zerlegen
- **Klarheit und Einfachheit:** Nicht zu viel Text und unnötige Ablenkungen, um den Fokus auf die Rechenaufgaben zu legen.

**Kinder mit ADHS** Kinder mit ADHS (Aufmerksamkeitsdefizit-/Hyperaktivitätsstörung)[1] haben Mühe die Aufmerksamkeitsspanne zu halten, leiden an starken Impulsivität und ausgeprägter körperlichen Unruhe. Um diese Kinder zu unterstützen, werden folgende Kriterien, bei der Gestaltung der Arbeitsblätter empfohlen. [12] [26]

- **Strukturierte Arbeitsblätter:** Arbeitsblätter in klar strukturierte Abschnitte einteilen. Nummerierte Aufgaben für eine klare Orientierung.
- **Kurze Aufgaben:** Um die Aufmerksamkeit besser zu halten, sollte die Arbeitsblätter kurze Aufgaben mit häufigen Pausen enthalten.
- **Visuelle Anker:** Visuelle Hinweise nutzen, wie Symbole, Farben oder Bilder, um wichtige Informationen hervorzuheben.
- **Begrenzte Ablenkungen:** Überladenen Designs und unnötige Grafiken nutzen, um die Konzentration nicht zu stören.

**Kinder mit ASS** "Autismus-Spektrum-Störungen sind tiefgreifende Entwicklungsstörungen, die unter anderem durch ein reduziertes Interesse an sozialen Kontakten sowie einem reduzierten Verständnis sozialer Situationen gekennzeichnet sind"[3]. Für diese Kinder gibt es wiederum weitere Kriterien für die Gestaltung der Arbeitsblätter, welche die Kinder wohlfühlen lassen und ihr Lernumfeld verbessern kann. [12] [26]

- **Routinen und Vorhersehbarkeit:** Gestaltung von Arbeitsblätter, nach klaren und vorhersehbaren Mustern. Kinder im Autismus-Spektrum fühlen sich wohler, wenn sie wissen was als Nächstes kommt.
- **Berücksichtigen von sensorischen Überempfindlichkeiten:** Möglichst schlichte, nicht überladene Arbeitsblätter verwenden, um sensorische Überforderung zu vermeiden.
- **Eindeutige und konkrete Anweisungen:** Präzise und klar strukturierte Anweisungen verwenden. Keine doppeldeutige Sprache oder Metaphern verwenden.

**Nutzen für diese Arbeit** Die sozialpädagogische Recherche hat zentrale Kriterien für die Gestaltung von Arbeitsblättern hervorgehoben, die beim Design des Prototyps berücksichtigt werden sollen. Einige Aspekte, wie das Vermeiden überladener Designs, sind nicht technisch überprüfbar und erfordern weiterhin die Einschätzung der Lehrperson. Der Prototyp wird jedoch so gestaltet, dass er die technischen Möglichkeiten zur Umsetzung dieser Kriterien bereitstellt. Dazu gehören die Anpassung von Schriftgrößen, Abständen, Farben und das Hinzufügen visueller Hilfsmittel, um die individuellen Bedürfnisse der Kinder optimal zu unterstützen.

### 3 Evaluierung der Umsetzungstrategie

Die zwei technischen Ansätze (siehe Abschnitt 2.1) wurden eingehend untersucht und bewertet, um eine geeignete Grundlage für die Umsetzung dieser Arbeit zu schaffen.

**Verzicht auf OCR-Technologie** Die Verwendung einer OCR-basierte Lösung zur Digitalisierung vorhandener Lehrmittel könnte eine solide Basis darstellen, damit Arbeitsblätter weiterverarbeitet und erweitert werden können. Die technischen Herausforderungen und Limitierungen der OCR-Technologie (siehe Abschnitt 2.1.1) wurden jedoch als kritisch eingestuft. Trotz der Fortschritte bei der Texterkennung bleiben Probleme wie schlechte Qualität der Bilder, komplexe Layouts und handgeschriebene Aufgaben bestehen. Eine präzise Erkennung von Aufgaben wäre nur mit erheblichem Aufwand umsetzbar und würde in keinem sinnvollen Verhältnis zum Nutzen für die Themenstellerin stehen.

Als möglicher Kompromiss könnte die Möglichkeit geboten werden, dass Arbeitsblättern mit vordefinierten Strukturen eingescannt werden könnten. Jedoch würde damit die Möglichkeit einer umfangreichen Anpassung und Weiterverarbeitung der Arbeitsblätter aus zeitlichen und technischen Gründen entfallen. Da dieser Nutzen für die Themenstellerin im Vordergrund steht, wurde entschieden, die Weiterentwicklung dieser Technologie in dieser Arbeit nicht weiterzuverfolgen.

**Fokus auf dynamische PDF-Generierung** Stattdessen wurde der Fokus auf die dynamische Generierung von Arbeitsblättern mithilfe von PDF-Generatoren gelegt. Dieses Konzept bietet eine effizientere, flexiblere und technisch umsetzbare Lösung innerhalb des vorgegebenen Zeitrahmens. Der Vorteil liegt darin, dass neue Arbeitsblätter mit klar definierten Layouts und angepassten Hilfestellungen generiert werden können, ohne die hohe Komplexität einer OCR-basierten Lösung zu benötigen. Somit sind die Lehrpersonen flexibel und können die Arbeitsblätter effizient an die Bedürfnisse der Schülerinnen und Schüler angepasst werden.

Der Fokus auf die Erstellung von digitalen Arbeitsblättern ermöglicht es, folgende Vorteile zu realisieren:

- **Strukturiertes Design:** Anpassbare Layouts und Vorlagen, die gestalterische und funktionale Aspekte (wie Schriften, Farben und Abstände) berücksichtigen.
- **Flexibilität:** Arbeitsblätter lassen sich dynamisch erstellen und individuell an die Bedürfnisse der Schülerinnen und Schüler anpassen.
- **Visuelle Unterstützung:** Einfügen einfacher Hilfsmittel wie Zahlenstrahlen, Tabellen oder Diagramme, die das Verständnis der Aufgaben fördern.
- **Effizienz:** Arbeitsblätter können direkt in ein PDF-Format umgewandelt und für den Druck oder die digitale Verbreitung verwendet werden.

**Entscheid PDF-Generator** Für die Tools iText7 und QuestPDF wurden zunächst kleine Proof of Concepts erstellt, um deren Funktionalität und Benutzerfreundlichkeit zu evaluieren. Das Hauptziel war es herauszufinden, welches Tool am besten für die Umsetzung dieser Arbeit geeignet ist. Der zentrale Aspekt war dabei die Möglichkeit, dynamische HTML-Inhalte in PDFs umzuwandeln.

Die FluentAPI von QuestPDF erwies sich als besonders benutzerfreundlich, da sie es ermöglicht, über verschiedene Objekte zu iterieren und dynamische PDFs zu erstellen. Allerdings konnte keine Lösung gefunden werden, um bestehende HTML-Komponenten direkt in PDFs einzubinden. Dies ist jedoch ein essenzielles Kriterium für diese Arbeit, da die Wiederverwendung von Komponenten sowohl in der Applikation als auch in den PDFs gewährleistet sein sollte.

Im Gegensatz dazu erfüllt iText7 dieses Kriterium: Es bietet eine Möglichkeit zur Einbindung von HTML-Komponenten in PDFs. Der durchgeführte Proof of Concept zeigte, dass die Umsetzung vergleichsweise unkompliziert ist. Aufgrund dieser Vorteile fiel die Entscheidung zugunsten von iText7 als PDF-Generator für die Arbeit.

## 4 Umsetzung des User-Centered Design

In diesem Kapitel wird auf das Vorgehen des User-Centered Designs im Rahmen dieser Arbeit eingegangen.

Da die Themenstellerin und deren Nutzen bei dieser Arbeit im Vordergrund steht, werden primär ihre Bedürfnisse und Erwartungen reflektiert. Dazu gab es während der Projektlaufzeit fortwährend Besprechungen, um den Entwurf des Produkts mit ihrer Vision abzugleichen.

Im ersten Teil werden die prototypischen Mockups vorgestellt und die Ergebnisse aus dem Austausch mit der Themenstellerin beschrieben.

Im zweiten Teil dieses Kapitels werden die Usability Tests beschrieben. Dazu wurden verschiedene Szenarien erstellt, um die Benutzerfreundlichkeit des Produkts zu erforschen.

### 4.1 Initiale Mockups

Nach den Erkenntnissen aus der Recherche und dem darauffolgenden Entscheid (Siehe Kapitel 3: Evaluierung der Umsetzungstrategie), wurden zwei Ideen entwickelt, welche als mögliche Implementierung vorstellbar waren:

- **Visuelle Gestaltung:** Einem Arbeitsblatt werden Kacheln hinzugefügt, welche einen Typ von Aufgabe, z.B. Rechenaufgabe repräsentieren. Diese Aufgaben werden direkt visuell in einem virtuellen Arbeitsblatt angezeigt.
- **Deklarative Gestaltung:** Die gewünschte Aufgaben können über Parameter definiert werden. Diese Liste an Aufgaben mit verschiedenen Parametern wird automatisch in ein druckbares Format überführt.

Die zwei Ideen wurden als Mockups visualisiert und der Themenstellerin vorgestellt, damit sie sich die Ideen besser vorstellen kann. Hierbei wurden die Vor- und Nachteile aufgezeigt und darüber diskutiert, welche Variante von der Themenstellerin bevorzugt wird.

Es soll unabhängig der Gestaltung die Möglichkeit geboten werden, dass die Darstellung im druckbaren Format individuell gestaltet werden kann. Für diesen Zweck wurde die Idee der Vorlagen entwickelt, welche eine Sammlung an Parametern darstellen, über welche die visuelle Darstellung der Aufgaben weiter verfeinert werden kann.

Auch sollen Arbeitsblätter in getrennte Bereiche aufgeteilt werden können um eine klare visuelle Trennung zwischen Aufgaben zu schaffen. Diese getrennten Bereiche wurden "Funktionsblöcke" getauft und beinhalten immer mindestens eine Aufgabe. Wenn zum Beispiel Additions und Subtraktionsaufgaben erstellt werden, diese aber klar voneinander getrennt sein sollen, dann könnten sie in verschiedene Funktionsblöcke eingeteilt werden.

#### 4.1.1 Grundlage

Einige Teile der Applikation ist unabhängig von der gewählten Variante. Dieser Rahmen wurde als erstes visualisiert.

Die Navigation wird für alle Seiten der Applikation sehr ähnlich aussehen, weshalb der Fokus zuerst darauf gelegt wurde. Drei Navigationspunkte wurden definiert:

- **Dashboard / Home:** Übersicht der Arbeitsblätter
- **Einstellungen:** Einstellungen der Applikation
- **Neues Arbeitsblatt:** Öffnet Dialog zur Erstellung eines neuen Arbeitsblatts. Die Idee war, jederzeit die Möglichkeit zu bieten ein neues Arbeitsblatt erstellen zu können.

Oben rechts stehen die Kürzel des eingeloggten Benutzers. Hierüber soll das Logout gesteuert werden.

Damit bereits erstellte Arbeitsblätter verwaltet werden können, soll es eine Ansicht geben, welche die erstellten Arbeitsblätter zugänglich macht. Auf dieser Ansicht sieht man die eigenen Arbeitsblätter und kann diese von hier aus wieder ausdrucken oder weiter bearbeiten.

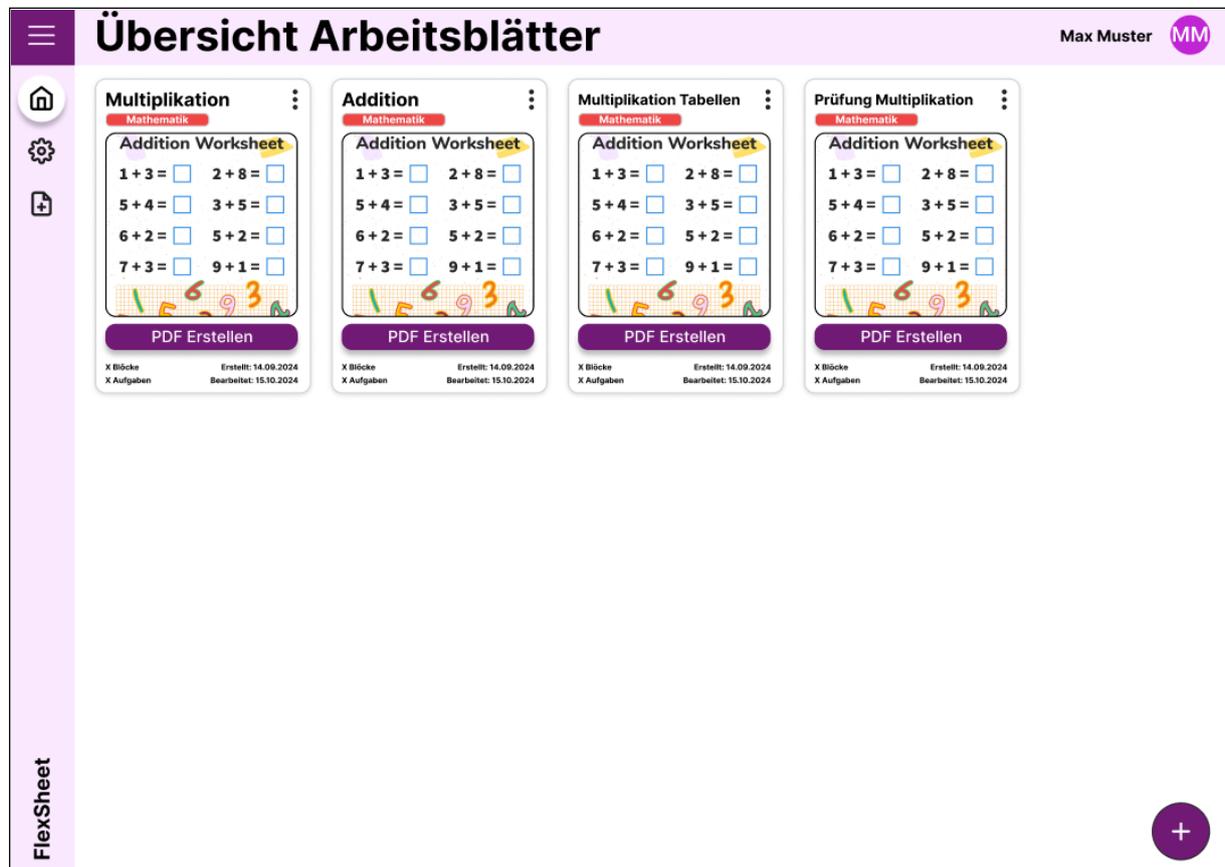


Abbildung 16: Wireframes Fundament: Dashboard

Quelle: *Erstellt mit Figma [14]*

Eine weitere Funktionalität, welche beide Varianten enthalten sollen, ist die Erstellung und die Bearbeitung der Vorlagen. Vorlagen dienen, wie schon angedeutet, der Speicherung von Parametern, welche die Gestaltung der Arbeitsblätter steuern. Diese Parameter sind zum Beispiel Schriftgrößen, Abstände, Anzahl Aufgaben etc. Die Stärke der Vorlagen liegt in der Wiederverwendbarkeit. Sind sie einmal sinnvoll definiert für die spezifische Bedürfnisse eines Kindes oder für eine Gruppe von Kindern, können Sie für jedes Arbeitsblatt erneut verwendet werden.

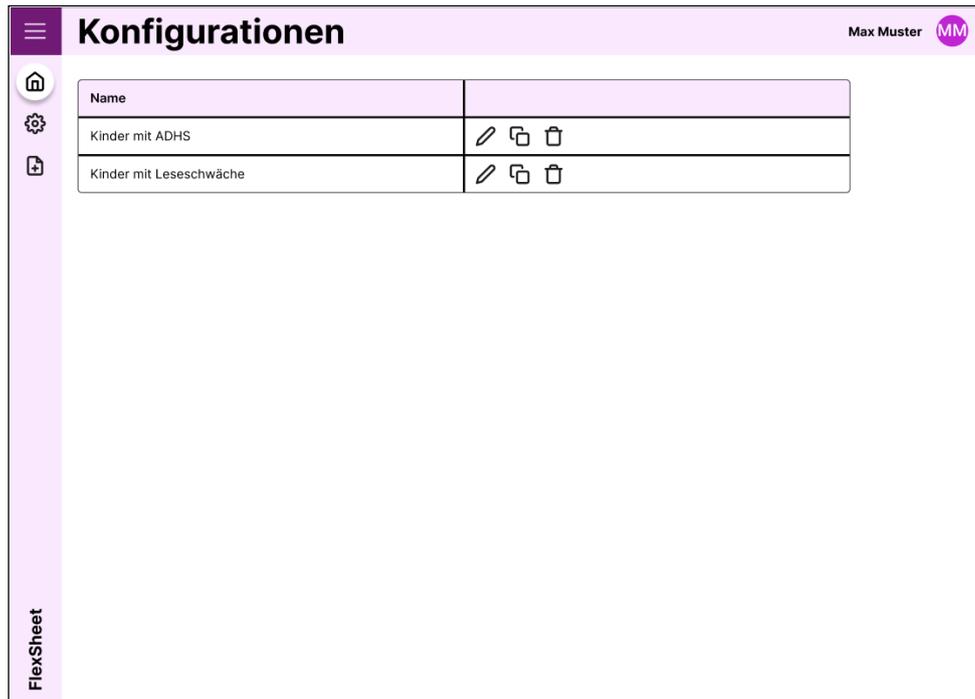


Abbildung 17: Wireframes Fundament: Vorlagen

Quelle: Erstellt mit Figma [14]

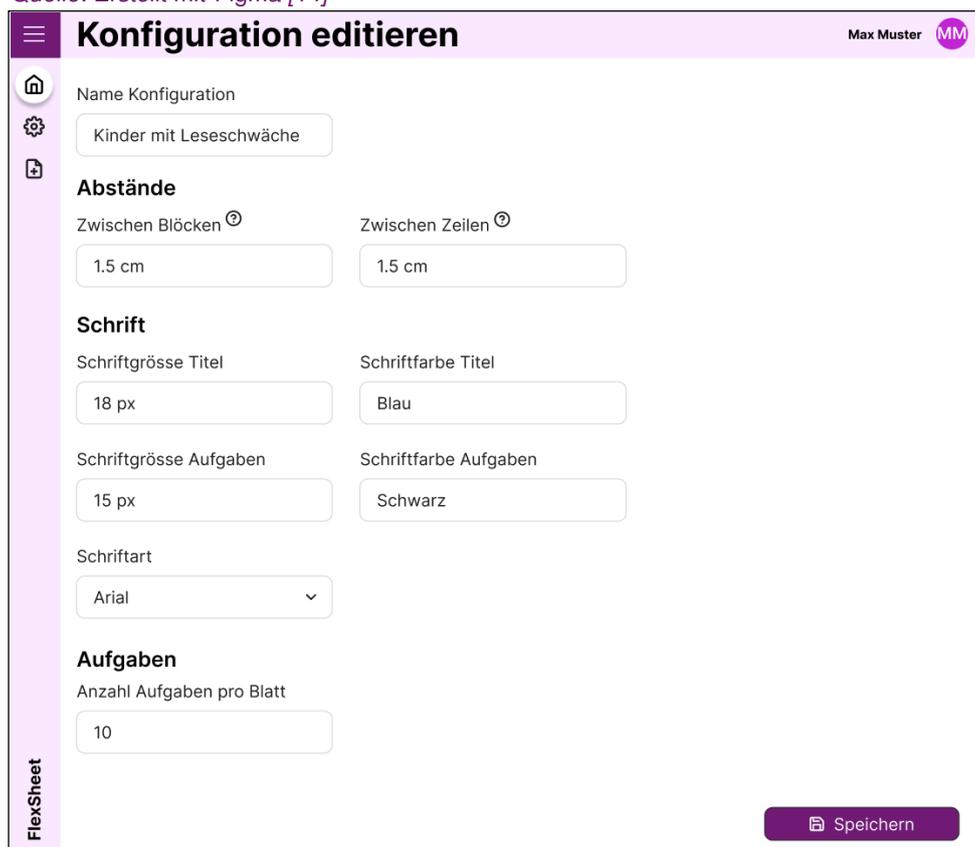


Abbildung 18: Wireframes Fundament: Vorlagen konfigurieren

Quelle: Erstellt mit Figma [14]

Beide Varianten erlauben zudem das Erstellen eines PDF. Die erstellten Arbeitsblätter definieren dabei die Menge und Typ und Inhalte der Aufgaben, während die Vorlagen zur Gestaltung des PDFs verwendet werden.

#### 4.1.2 Variante 1: Manuelles Arbeitsblatt

Diese Idee entstand aus dem Bedürfnis, dass Nutzer\*innen mit wenig technischer Erfahrung möglichst schnelles visuelles Feedback erhalten sollen. Dafür bietet sich ein Baukasten-System sehr gut an.

Somit wurde in dieser ersten Variante der Fokus auf die manuelle Verwaltung des Layouts gesetzt. Der Startpunkt eines neuen Arbeitsblattes ist eine leere Seite mit einem Titel. Dieses Blatt kann dann ausgebaut werden, indem Funktionsblöcke erstellt werden, welche die Seite in mehrere Bereiche unterteilt. Im unteren Bereich des Arbeitsblattes könnten auch weitere Seiten hinzugefügt werden. Auf der rechten Seite sind verschiedene Parameter verfügbar, über welche das Basis-Layout genauer definiert werden kann.

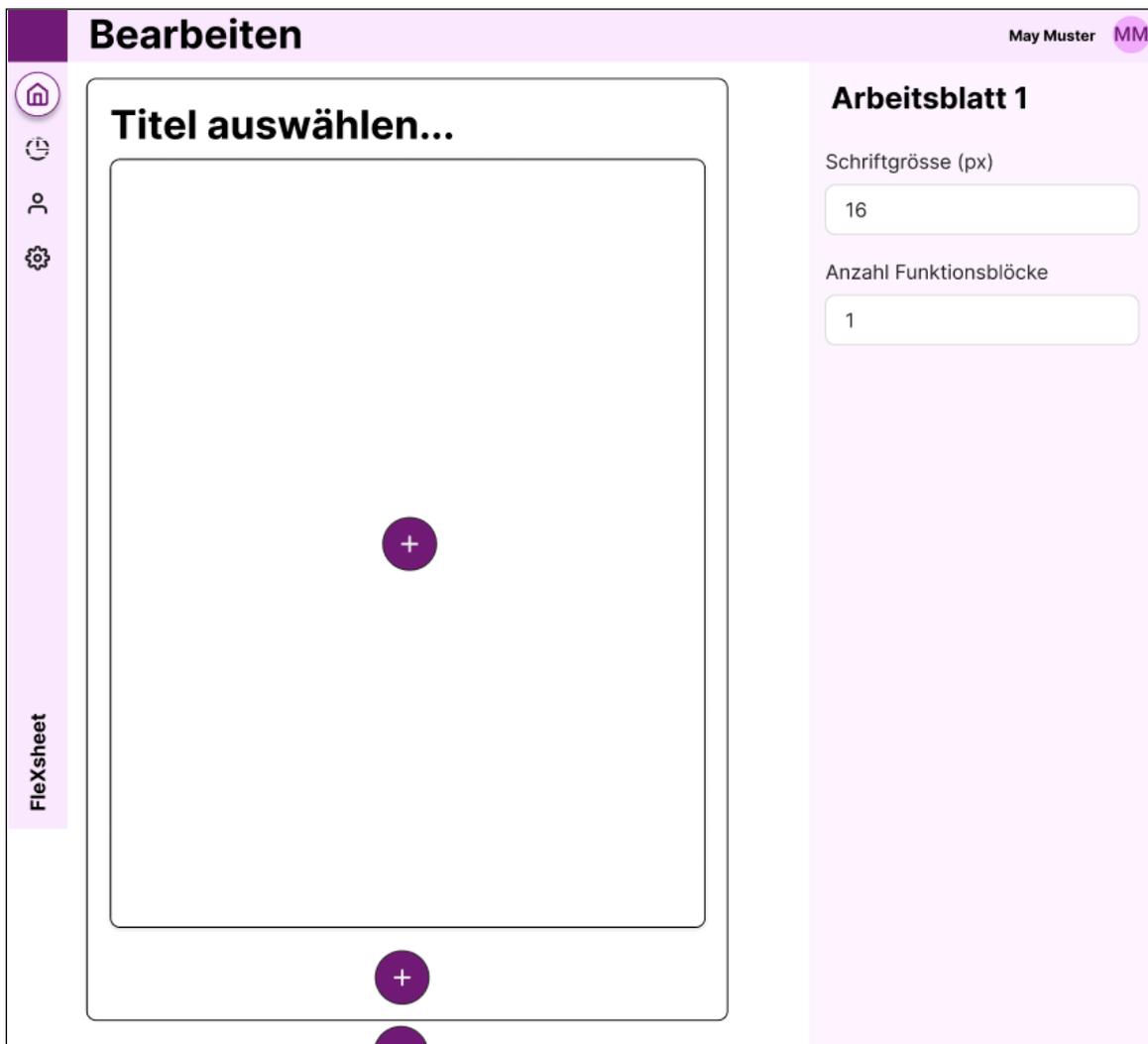


Abbildung 19: Wireframes Variante 1: Layout  
Quelle: Erstellt mit Figma [14]

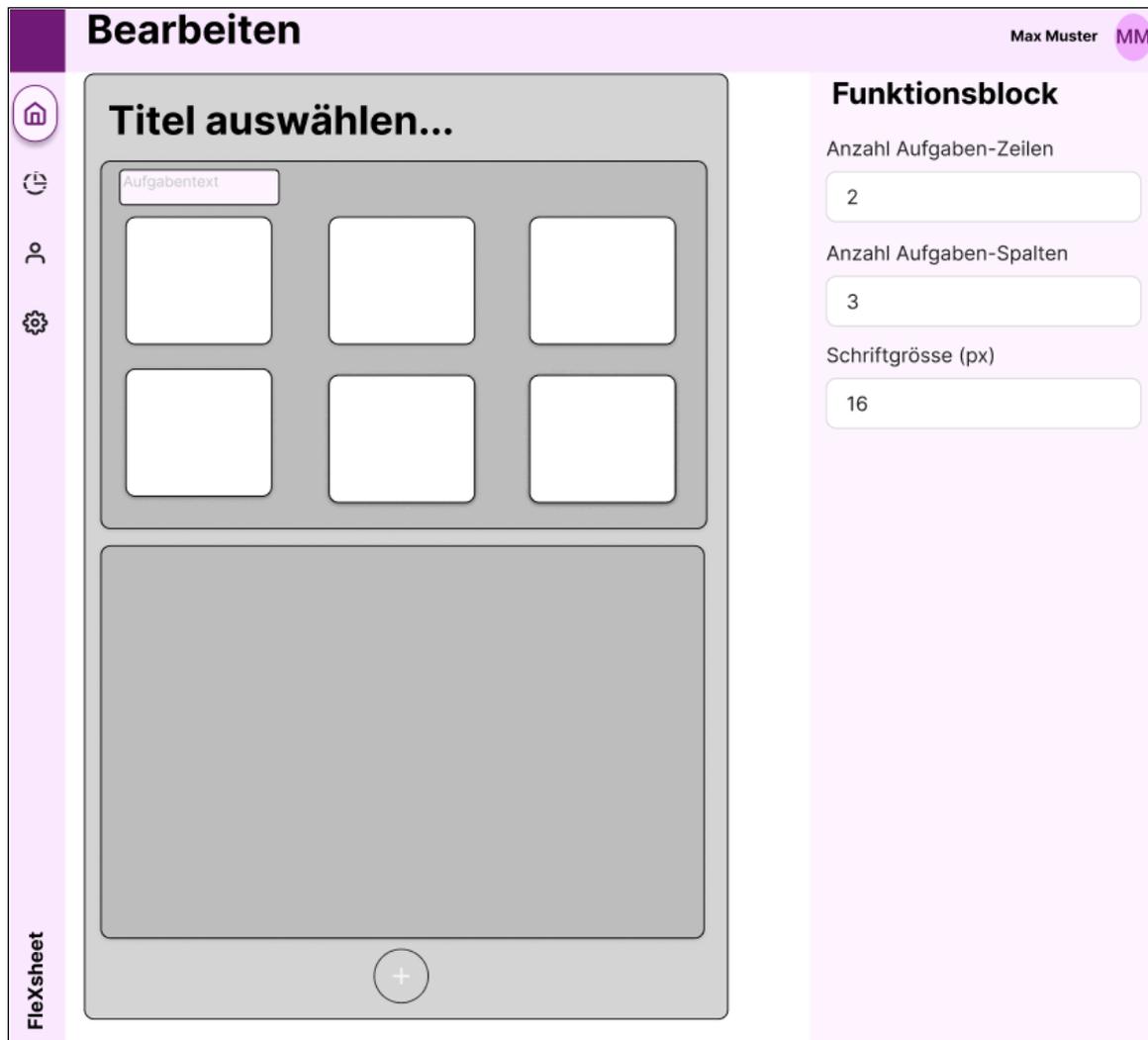


Abbildung 20: Wireframes Variante 1: Blöcke verwalten

Quelle: *Erstellt mit Figma [14]*

Ein Funktionsblock wurde hierbei definiert, als die erste Hierarchiestufe des Arbeitsblattes. Es teilt eine Seite in mehrere Abschnitte und kann so visuell abgetrennt werden, durch Trennstri- che oder Rahmen. Ein Funktionsblock kann einen Titel enthalten, welcher zum Beispiel die Überschrift für eine Gruppe von Aufgaben sein kann. Die einzelnen Funktionsblöcke können dann individuell mit Grössenangaben skaliert werden. In einem Funktionsblock können meh- rere Aufgabenblöcke enthalten sein. Ein Funktionsblock kann in mehrere Zeilen und Spalten unterteilt werden, wobei eine Zelle einen Aufgabenblock repräsentiert.

Jeder Aufgabenblock hat jeweils einen Aufgabentyp. Dieser beschreibt die Art von Mathemati- kaufgabe welche dargestellt werden soll:

- Multiplikation
- Division
- Addition
- Subtraktion
- Zahlenstrahl
- 100er Feld (Tabelle)



Abbildung 21: Wireframes Variante 1: Aufgaben hinzufügen

Quelle: *Erstellt mit Figma [14]*

In einem Aufgabenblock können mehrere Aufgabentypen zusammengestellt und definiert werden. Jeder Aufgabenblock kann eine gewisse Anzahl Aufgaben unterbringen, sie müssen jedoch nicht alle gleich viele Aufgaben beinhalten und auch nicht vom gleichen Typ sein. Als Beispiel könnten in einem Aufgabenblock drei Multiplikationsaufgaben definiert werden und in nächsten zwei Divisionsaufgaben. Dies sollte garantieren, dass die Lehrpersonen die Arbeitsblätter möglichst uneingeschränkt erstellen können.

### Vorteile

- Das Layout des Arbeitsblatt kann individuell nach den Bedürfnissen angepasst werden und bietet die volle Kontrolle.
- Es kann genau definiert werden, wo welche Aufgaben liegen sollen und sie können nach Belieben ausgetauscht werden.
- Die Gestalterischen Elemente, können für jedes Element beliebig geändert werden.
- Direktes visuelles Feedback hilft bei der Erstellung.

### Nachteile

- Aufbau des Layout sehr zeitintensiv, da sich viele Gedanken zu der genauen Struktur gemacht werden müssen.

- Manuelle Definition der Werte jeder einzelnen Aufgabe (innerhalb eines Aufgabenblocks). Das bedeutet, dass die Zahlen innerhalb der Aufgaben von Hand eingetragen werden müssen.
- Darstellung des PDFs darf nicht zu stark von der visuellen Repräsentation abweichen. Ansonsten ist der Vorteil des direkten Feedbacks weniger nützlich.
- Komplexe Berechnungen der Layouts: Innerhalb eines Aufgabenblocks muss die Grösse der Aufgaben berechnet werden, damit diese überhaupt korrekt dargestellt werden können. Anhand dieser Berechnungen würde dann eine maximale Anzahl Spalten definiert werden. Benutzer müssten somit oft die Parameter des Funktionsblocks anpassen, sobald ein Aufgabenblock erstellt wird.

#### **4.1.3 Variante 2: Generiertes Arbeitsblatt**

Die zweite Variante verfolgt einen deklarativen Ansatz der Arbeitsblatterstellung. Die Idee dahinter war, dass Nutzerinnen und Nutzer schnell ein Arbeitsblatt erstellen können, ohne sich um die genauen Details kümmern zu müssen. Die zeitaufwendige Arbeit der Gestaltung soll von der Applikation übernommen werden, während sich die Lehrpersonen auf den Inhalt fokussieren können.

Dafür wird beim Erstellen eines Arbeitsblattes eine Konfigurationsseite aufgerufen. Diese Ansicht bietet die Möglichkeit allgemeine Parameter zu definieren und eine Vorlage auszuwählen.

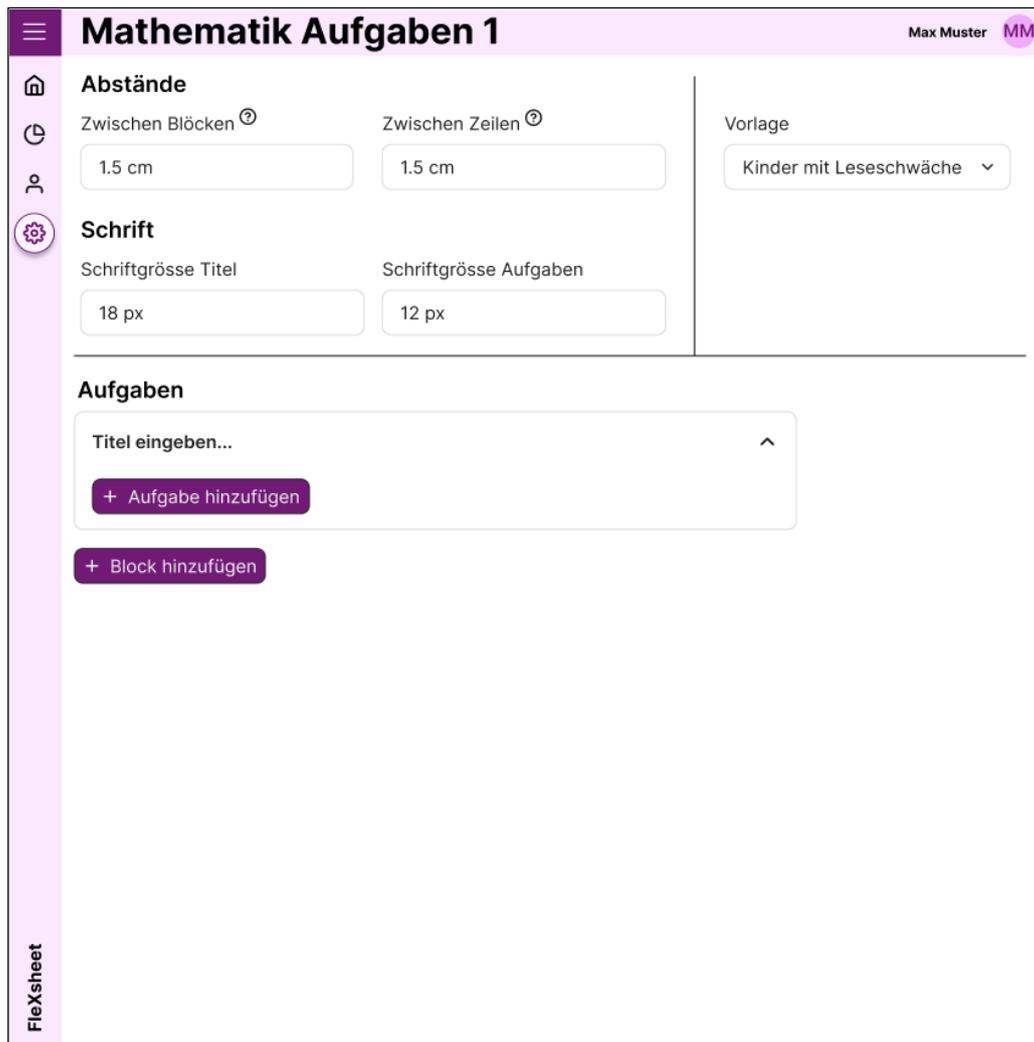


Abbildung 22: Wireframes Variante 2: Layout

Quelle: *Erstellt mit Figma [14]*

Im unteren Teil der Konfigurationsseite befindet sich die Aufgabenaufstellung. Die Unterteilung in Blöcke und Aufgaben besteht auch bei dieser Variante. Die Blöcke und Aufgaben werden jedoch nicht direkt visuell angezeigt, sondern sind als Liste ersichtlich. Beim Hinzufügen von Aufgaben kann der Typ, Anzahl und Zahlenbereich angepasst werden. Die definierten Aufgaben sind dann in der Liste ersichtlich und man kann auf einen Blick erkennen, was diese beinhalten.

**Mathematik Aufgaben 1** Max Muster

**Abstände**

Zwischen Blöcken 1.5 cm

Zwischen Zeilen 1.5 cm

Vorlage: Kinder mit Leseschwäche

**Schrift**

Schriftgröße Titel: 18 px

**Aufgaben**

Titel eingeben...

+ Aufgabe hinzufügen

+ Block hinzufügen

**Aufgabe definieren**

Typ: Rechenaufgabe

Operation: Multiplikation

Anzahl Aufgaben: 10

Zahlenbereich von - bis: 1 - 10

Aufgabe speichern

Abbildung 23: Wireframes Variante 2: Aufgaben definieren  
Quelle: Erstellt mit Figma [14]

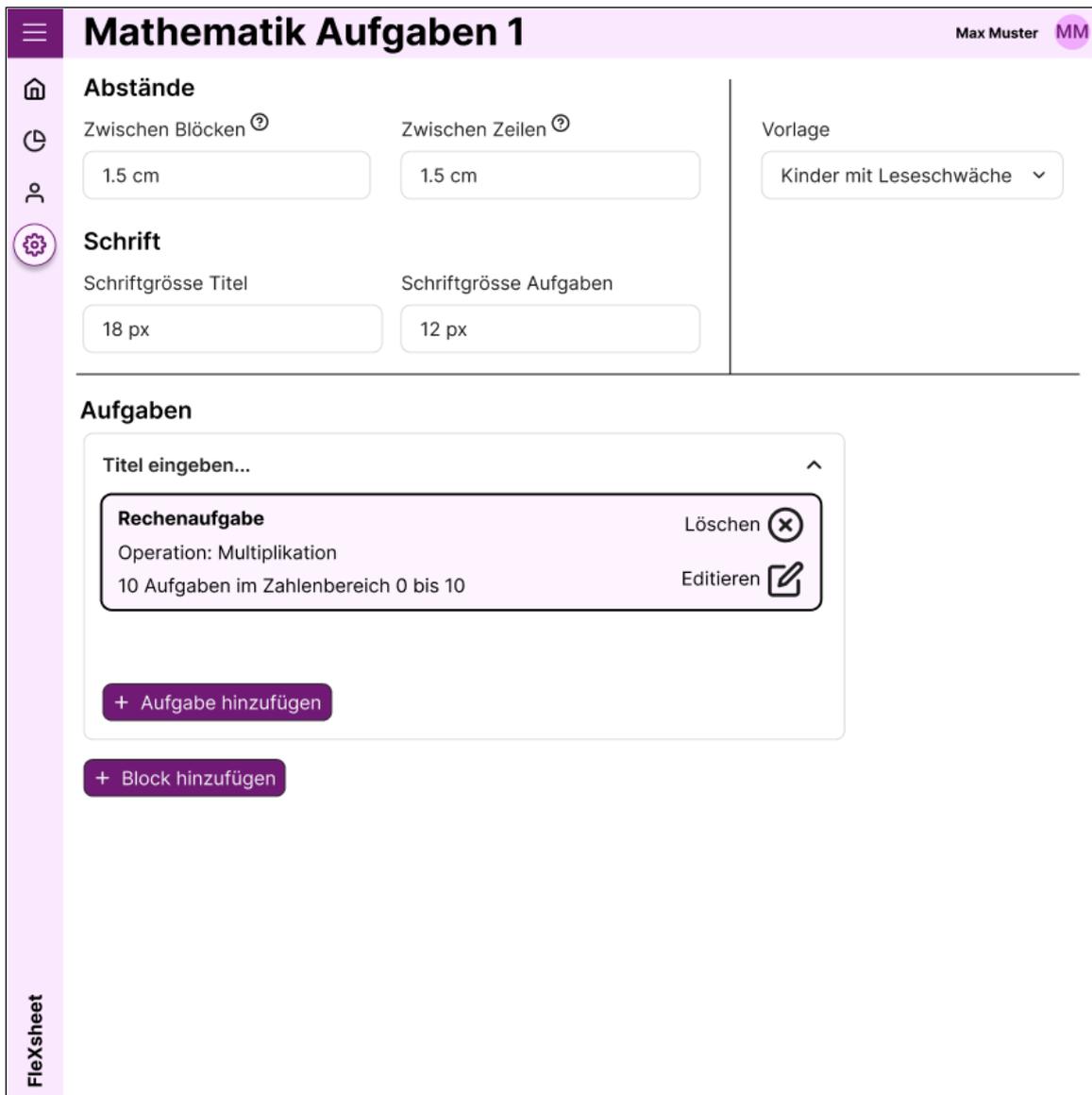


Abbildung 24: Wireframes Variante 2: Aufgaben werden hinzugefügt

Quelle: *Erstellt mit Figma [14]*

Nachdem man mit der Konfiguration des Arbeitsblattes fertig ist, kann man sich daraus ein Arbeitsblatt generieren lassen. Das System achtet selber auf eine sinnvolle Strukturierung des Arbeitsblattes anhand der definierten Aufgaben.

### Vorteile

- Lehrpersonen können sich auf die Inhalte fokussieren ohne sich Gedanken zum genauen Layout machen zu müssen.
- Arbeitsblätter können sehr schnell generiert werden.
- Visuelle Fehler werden auf ein Minimum reduziert, da die Layout-Gestaltung automatisch vorgenommen wird.

### Nachteile

- Lehrpersonen können nur über die Parameter der Vorlagen Einfluss auf das Layout nehmen.
- Sehr spezifische Anforderungen können nicht umgesetzt werden.

## 4.2 Ausarbeiten der Anforderungen

### 4.2.1 Erster Austausch mit der Themenstellerin

Die Vor- und Nachteile der beiden Varianten wurden in einem ersten Treffen nach dem Kick-off ausgiebig mit der Themenstellerin diskutiert. Dabei wurden ihr die Mockups vorgestellt und unsere Vision für die jeweiligen Varianten erklärt. Sie hat sich dann für die Variante 2 entschieden, da sie die schnelle und unkomplizierte Erstellung von Arbeitsblättern als besonders vorteilhaft empfindet. Die Einfachheit dieser Variante war für sie ein ausschlaggebendes Argument, da sie so zeitsparend arbeiten kann.

Auch aus unserer Perspektive wurde diese Entscheidung unterstützt, da die schnelle und effiziente Erstellung von Arbeitsblättern im Fokus der Arbeit steht. Bei Variante 2 können sich Lehrpersonen auf den Inhalt konzentrieren, ohne sich um die Gestaltung oder spezifischen Zahlen der Aufgaben kümmern zu müssen.

Danach wurde festgelegt, dass die Themenstellerin eine Liste mit Kriterien für die Vorlagen erstellt. Diese sollten die wichtigsten Eigenschaften eines Arbeitsblattes aufzeigen, welche nach Lernschwächen gruppiert sind. Somit können die benötigten Parameter für die Vorlagen definiert werden. Parameter welche den Rahmen dieser Arbeit sprengen würden, können als Erweiterungsmöglichkeiten notiert werden.

Sonderpädagogische Bedürfnisse und Kategorien für die Anpassung von Arbeitsblättern aus Lehrmitteln (Mathematik)							
	Anzahl Aufgaben	Einfache Sprache	Layout / Struktur	Visualisierung	Belohnung (Bild, Aussage)	Übersetzung	Schrift
Autismus	x	x	x	x			x
Lernbehinderungen	x	x	x	x			
Sprach- und Sprechstörungen	x	x	x	x			
ADHS	x		x	x	x		x
Emotionale und Verhaltensstörungen	x		x	x	x		
Körperliche Beeinträchtigungen (z.B. CP)			x	x			x
Lese- Rechtschreibstörung (LRS)		x	x	x			*x
Rechenschwäche	x		x	x	x		
Hochbegabung	x	x	x	x	x		
Deutsch als Zweitsprache	x	x	x	x		x	*x

**Kategorien für die digitale Version:**

- Audiovisuelle Unterstützung (Sprachsteuerung, Vorlesefunktion)
- Haptisches Feedback (z.B. Vibration, wenn die Aufgabe beendet ist)
- Interaktive Elemente (z.B. Visualisierung manipulieren – Hunderterfeld färben usw.)

\*Aufgabenstellungen in Silbenschrift (rot/blau). Beispiel: **Wie viele...**

Abbildung 25: Sonderpädagogische Bedürfnisse

Quelle: *Erstellt von Themenstellerin [34]*

#### 4.2.2 Teamreflexion

Nach dem wichtigen Gespräch mit der Themenstellerin, wurde im Team nochmals das Datenmodell und der Programmfluss hinter den initialen Mockups analysiert. Bei der Variante 2 (Siehe Kapitel 4.1.3: Variante 2: Generiertes Arbeitsblatt), sah der Programmfluss für das Erstellen von Arbeitsblättern folgendermassen aus:

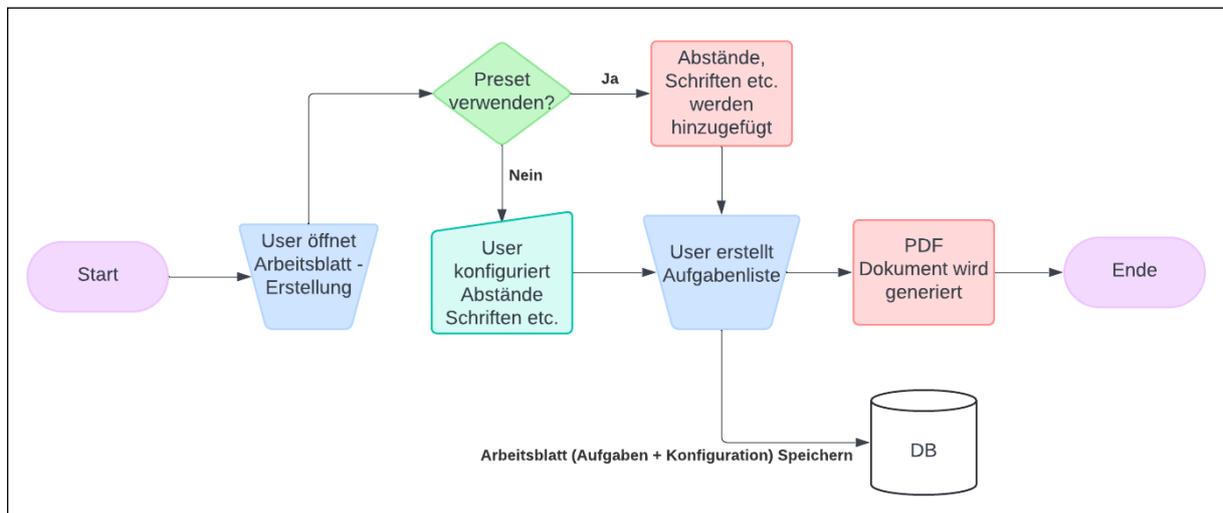


Abbildung 26: Programmfluss Variante 2

Quelle: Erstellt mit Lucidchart [22]

Hierbei ist aufgefallen, dass der Prozess effizienter gestaltet werden kann. Die Verknüpfung von Layout-Konfigurationen (Abstände, Schriften etc.) und Aufgaben-Deklarationen macht wenig Sinn. Nehmen wir an, eine Lehrperson definiert ein Arbeitsblatt, möchte jedoch für verschiedene Schüler verschiedene Zeilenabstände machen. Die starke Kopplung führt dann dazu, dass für jeden Zeilenabstand ein neues Arbeitsblatt erstellt werden müsste, was gegen unsere Vision verstösst. Es würde mehr Sinn machen, dass die Lehrperson eine Aufgabenliste zusammenstellen kann, welche sie abspeichern und weiter bearbeiten kann. Diese Aufgabenliste kann sie dann, basierend auf den definierten Vorlagen oder individuellen Konfigurationen, in ein PDF umwandeln. Somit kann sie die Aufgabenliste immer wieder verwenden und sich ein neues Layout anhand der Vorlagen oder manuellen Konfigurationen generieren lassen.

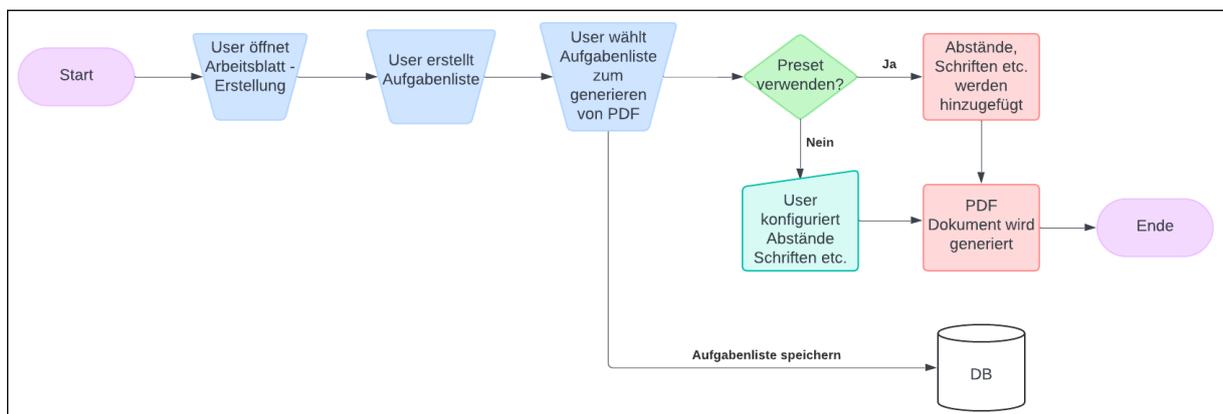


Abbildung 27: Programmfluss Überarbeitet

Quelle: Erstellt mit Lucidchart [22]

Nach diesem Austausch wurden die Mockups anhand der Variante 2 und der Anpassung des

Programmflusses ganz umfänglich fertiggestellt. Die kompletten Mockups sind unter dem Kapitel B: Mockups zu finden.

The image shows a mobile application interface for creating a new worksheet. The title bar at the top is purple and contains the text 'Neues Arbeitsblatt' on the left and 'Max Muster MM' on the right. A vertical sidebar on the left contains icons for home, settings, and a plus sign. The main content area has a purple header with a back arrow and the text 'Welchen Namen soll das Arbeitsblatt erhalten?'. Below this is a red-bordered text input field with the placeholder 'Titel eingeben...' and a red error message 'Dies ist ein Pflichtfeld' underneath. A section titled 'Aufgaben' (Tasks) follows, containing another text input field with the placeholder 'Titel eingeben...' and a small upward arrow icon. Below the input field is a purple button with a plus sign and the text '+ Aufgabe hinzufügen'. At the bottom of the main content area is another purple button with a plus sign and the text '+ Block hinzufügen'. The 'FlexSheet' logo is visible in the bottom left corner of the app frame.

Abbildung 28: Erstellen eines Arbeitsblatt nach Überarbeitung  
Quelle: *Erstellt mit Figma [14]*

### 4.3 Zweiter Austausch mit der Themenstellerin

Nach Fertigstellung der überarbeiteten Mockups, sowie Definition der funktionalen und nicht funktionalen Anforderungen, wurde ein letzter Austausch vor dem Start der Implementierung geplant. Das Ziel war nochmals Details zu besprechen und sicherzustellen, dass unsere Vorstellung mit der der Themenstellerin übereinstimmt. Bei diesem Gespräch wurden die überarbeiteten Mockups nochmals vorgestellt und mit ihr besprochen. Die Themenstellerin legte von Beginn an viel Wert auf die Visualisierungen der Rechenaufgaben, weshalb diese im Detail besprochen wurden. Eine wichtige Erkenntnis war, dass die geplanten Visualisierungen nicht ganz mit dem übereinstimmen, was die Schüler kennen.

Ursprünglich war vorgesehen, dass eine Multiplikationsaufgabe in Form einer Tabelle visualisiert werden kann. Die Kinder in der Schule sind sich bei Multiplikationen jedoch an Visualisierungen mit Wendepüttchen gewöhnt:

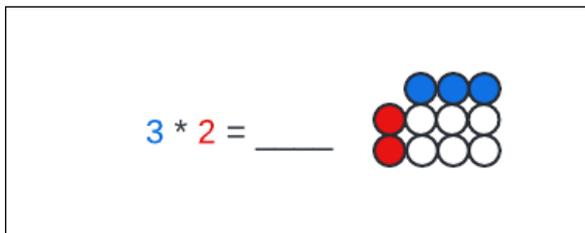


Abbildung 29: Multiplikationsaufgabe als Wendepüttchen visualisiert

Quelle: *Erstellt mit Lucidchart [22]*

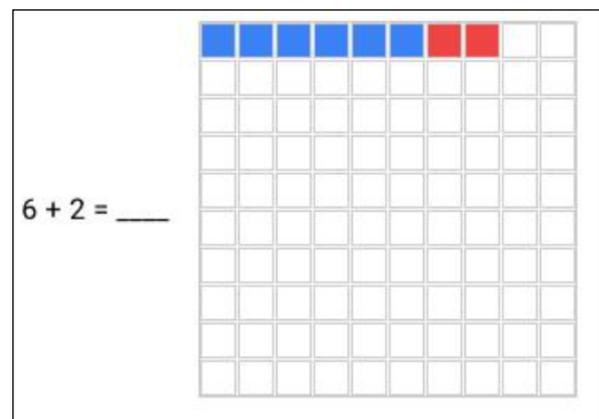


Abbildung 30: Additionsaufgabe als Tabelle visualisiert

Quelle: *Erstellt mit Lucidchart [22]*

Die Visualisierung als Tabellenform soll momentan nur bei Additionsaufgaben genutzt werden. Dieses Missverständnis wurde in den funktionalen Anforderungen angepasst und entsprechend während der Implementierung umgesetzt.

## 4.4 Usability Test

Usability-Tests sind ein wichtiger Bestandteil des menschenzentrierten Designs und helfen dabei, die Benutzerfreundlichkeit eines Produkts zu reflektieren. Sie sollen dabei helfen zu verstehen, welche Abläufe klar sind und wo noch Schwierigkeiten bestehen. Zusätzlich sollen die Testpersonen die Applikation aus einem anderen Blickwinkel betrachten um Ungereimtheiten und Fehler in der Applikations-Logik zu identifizieren.

Als Vorbereitung für die Tests wurde ein detaillierter Plan entwickelt, der die Testziele und die angewandte Testmethodik umfasst. Diese Schritte sind entscheidend, um sicherzustellen, dass die gesammelten Daten aussagekräftig sind. Die Resultate der Usability-Tests sind im Kapitel 9.2: Usability Test - Testergebnisse festgehalten.

### 4.4.1 Testziele

Um die gewünschten Erkenntnisse aus den Usability-Tests zu gewinnen, wurden klare Testziele festgelegt, welche bei der Gestaltung der Test-Szenarien als Leitfaden gelten sollen.

Die Hauptziele umfassen:

- **Identifikation von Usability-Problemen:** Erkennen, wo Benutzer auf Schwierigkeiten stossen oder ineffiziente Abläufe feststellen.
- **Effizienz der Nutzung:** Überprüfen, wie schnell und effektiv die Benutzer ihre Aufgaben durchführen können.
- **Feedback zur Nutzererfahrung:** Sammeln von Eindrücken und Meinungen der Benutzer zur Gestaltung und Funktionalität des Produkts.
- **Feedback zur Erweiterungsmöglichkeiten:** Identifizierung von gewünschten Erweiterungsmöglichkeiten

### 4.4.2 Testmethodik

Die Testmethodik umfasst eine Kombination aus qualitativen und quantitativen Ansätzen, um ein ganzheitliches Bild der Nutzererfahrung zu erhalten. Durch diese Methodenkombination können Erkenntnisse gewonnen werden, die als Grundlage für zukünftige Verbesserungen des Designs dienen. Die wichtigsten Methoden beinhalten:

- **Beobachtungen:** Während den Tests werden die Teilnehmer bei der Nutzung des Produkts beobachtet, um direkt zu sehen, wo Schwierigkeiten auftreten.
- **Nachbesprechung:** Nach den Tests werden die Teilnehmer befragt, um mehr in ihre Erfahrungen und Meinungen zu erhalten.
- **Think-Aloud-Protokolle:** Teilnehmer werden instruiert ihre Gedanken während der Nutzung verbal zu äussern, um Einsichten in ihre Entscheidungsprozesse und Reaktionen zu gewinnen.

### 4.4.3 Szenarien

Die Szenarien wurden so ausgelegt, dass sie möglichst alle Funktionalen Anforderungen (Siehe Kapitel 5.1: Funktionale Anforderungen) abdecken und zugleich die Testziele widerspiegeln können. Dazu wurden Szenarien erstellt, welche die Teilnehmer\*innen realitätsnah durch die gesamte Applikation leiten, jedoch genügend Abstrakt gestaltet sind, dass die Teilnehmer\*innen sich selber mit der Benutzung der Applikation auseinandersetzen müssen.

**Szenario 1 - Neue Schüler mit Lernschwächen** In deiner Klasse sind seit dem Sommer zwei neue Kinder dazugekommen. Diese Kinder benötigen besondere Unterstützung, da sie mit Lernschwächen zu kämpfen haben. Da du dich sehr um diese Kinder sorgst, ist es dir wichtig die Arbeitsblätter so zu gestalten, dass die Kinder besser damit arbeiten können.

Jonathan leidet unter einer starken Leseschwäche, ist ein extrem intelligentes Kind, doch kommt nicht einfach weiter, da er sich zu sehr beschäftigen muss, die Arbeitsblätter korrekt zu lesen. Paula leidet unter einer IQ-Schwäche und fühlt sich dadurch schnell überfordert und kommt im Gegensatz zu den anderen Kindern nicht so schnell vorwärts.

Damit du diese Kinder unterstützen kannst, möchtest du die Arbeitsblätter auf ihre Bedürfnisse anpassen können. Hierfür wäre es für dich am einfachsten, wenn du für beide Kinder ein Profil anlegen könntest, welches dich bei der Erstellung der Arbeitsblätter unterstützen kann.

**Szenario 2 - Neues Aufgabenblatt** Die Kinder in deiner Klasse sind im Gegensatz zu anderen Kindern aus der Schule nicht so stark im Fach Mathematik. Als Unterstützung möchtest du den Kindern jeden morgen ein kleines Quiz geben, welches die elementaren Rechenaufgaben testet, aber nicht benotet wird. Mit der Wiederholung dieses Quiz hoffst du, dass die Kinder ihre Kompetenzen in der Mathematik schnell verbessern können, ohne dass sie den Druck einer Prüfung spüren müssen. Dieses Quiz soll einfachere Rechnungen beinhalten, aber auch welche die knifflig sind.

In deiner Klasse gibt es jedoch auch Kinder, für welche ein solches Quiz unlösbar ist. Du weisst aber, dass diese Kinder erheblich von grafischen Darstellungen der Aufgaben profitieren würden. Für diese Kinder möchtest du ein eigenes Arbeitsblatt zusammenstellen, welches die elementaren Rechenaufgaben grafisch so repräsentiert, dass sie es besser verstehen können.

**Szenario 3 - Generieren von verschiedenen PDF** Die nächste Mathematiklektion steht vor der Tür, und du hast zwei tolle Aufgabensammlungen erstellt, welche auf die Bedürfnisse deiner Schüler zugeschnitten ist. Nun möchtest du sicherstellen, dass du alles bereit hast, um den Unterricht reibungslos durchführen zu können. Ein gedrucktes Exemplar wäre dabei besonders hilfreich. Mit einem solchen Arbeitsblatt könntest du die Aufgaben den Schülern direkt vorlegen und ihnen den Zugang zu den Übungen erleichtern. Es wäre grossartig, wenn du das Aufgabenblatt in ein druckbares Format bringen könntest, das du einfach mitnehmen kannst.

Dabei möchtest du aber auch darauf achten, dass Jonathan und Paula nicht zu kurz kommen.

**Szenario 4 - Editieren der Aufgabensammlung** Nach dem Durchsehen des erstellten Arbeitsblattes fällt dir ein kleiner Fehler auf – vielleicht ein Zahlendreher oder eine ungenaue Formulierung. Ausserdem merkst du, dass eine der Aufgaben nicht ganz deinen ursprünglichen Vorstellungen entspricht. Du fragst dich, ob du die Zahlenbereiche oder vielleicht auch die Rechenoperatoren anpassen könntest, damit die Aufgaben besser auf die Fähigkeiten deiner Schüler abgestimmt sind. Es wäre ideal, wenn du diese Änderungen flexibel vornehmen und das Arbeitsblatt mit den neuen Inhalten abspeichern könntest, bevor du es erneut verwendest.

## 5 Anforderungen

Nach der Analyse der Ausgangslage und dem Austausch mit der Themenstellerin, wurden die Bedürfnisse und Erwartungen an diese Arbeit in Funktionale Anforderungen für den Prototypen umgewandelt. Zusätzlich wurden Nicht-funktionale Anforderungen definiert, welche für eine moderne Webapplikation von zentraler Bedeutung sind und vom Entwicklerteam als essenziell eingestuft wurden. Diese Anforderungen stellen sicher, dass die Applikation nicht nur den aktuellen Bedürfnissen entspricht, sondern auch eine solide Grundlage für zukünftige Weiterentwicklungen bietet.

### 5.1 Funktionale Anforderungen

Die Funktionalen Anforderungen wurden aufgeteilt in Primary- und Secondary Features. Die Primary Features legen fest, was das definierte MVP ist. Diese Funktionen sollen auf jeden Fall innerhalb der Projektzeit implementiert werden. Die Secondary Features zeigen auf, welche Anforderungen bereits diskutiert wurden und bei genügend Zeit noch in das Projekt mit einfließen werden.

Die Funktionalen Anforderungen werden in Form von User Stories festgehalten, wobei die Themenstellerin als SHP repräsentiert wird.

#### 5.1.1 Primary Features

<b>ID</b>	US-1
<b>Bezeichnung</b>	Erstellen von Aufgabensammlungen
<b>Beschreibung</b>	Als SHP möchte ich die Möglichkeit haben verschiedene Aufgabensammlungen erstellen zu können. Die Aufgabensammlungen möchte ich als Basis für die Erstellung von Aufgabenblätter benutzen, weshalb sie wiederwendbar sein sollen. Die Einzelnen Aufgaben möchte ich innerhalb von Blöcken gruppieren können, wobei ein Block mehrere Aufgaben enthalten kann.

Tabelle 1: Functional Requirements (US-1)

#### Akzeptanzkriterien (US-1)

- Auf der Seite zum Erstellen von Aufgabensammlungen, kann man Blöcke und Aufgaben in die Sammlung hinzufügen, sie editieren, sowie entfernen.
- Die einzelnen Aufgaben sind konfigurierbar und werden in einer Übersicht dargestellt.
- Den einzelnen Blöcken kann man eine Überschrift zuweisen (z.B. Aufgabe 1: Rechenaus.)
- Die Aufgabensammlungen können gespeichert werden.

<b>ID</b>	US-2
<b>Bezeichnung</b>	Verwalten von Aufgabensammlungen
<b>Beschreibung</b>	Als SHP möchte ich eine Übersicht über alle erstellten Aufgabensammlungen haben. Diese möchte ich bei Bedarf umbenennen, löschen, oder editieren können.

Tabelle 2: Functional Requirements (US-2)

### Akzeptanzkriterien (US-2)

- Es gibt eine Seite mit der Übersicht von allen bereits erstellten Aufgabensammlungen.
- Die Aufgabensammlungen sind beschriftet mit Titel, Erstellungsdatum und Fachbezeichnung
- Die einzelnen Aufgabensammlungen können umbenannt, gelöscht, oder editiert werden.

<b>ID</b>	US-3
<b>Bezeichnung</b>	Typen von Aufgaben 1
<b>Beschreibung</b>	Als SHP möchte ich verschiedene Aufgabentypen der Mathematik zu einer Aufgabensammlung hinzufügen können. Ich möchte mich nicht darum kümmern einzelne aufgaben auszufüllen, sondern möchte, dass das System mir beliebig viele Aufgaben generieren kann. Da ich Schüler mit Lernschwächen betreue, möchte ich für gewisse Aufgaben sinnvolle Visualisierungen hinzufügen können, um die Schüler zu unterstützen. Ausserdem möchte ich manchmal verschiedene Auffrischungsaufgaben oder Belohnungen auf den Aufgabenblätter hinzufügen. Diese möchte ich als Bilder oder Text hinzufügen können.

Tabelle 3: Functional Requirements (US-3)

### Akzeptanzkriterien (US-3)

- In der Aufgabenerstellung kann man Multiplikations-, Divisions-, Additions und Subtraktionsaufgaben hinzufügen. Diese sollen mit folgenden Parametern generiert werden:
  - Anzahl Aufgaben
  - Zahlenbereich der Aufgaben (Lösungsbereich)
  - Bei der Konfiguration der Aufgaben hat man die Möglichkeit eine Visualisierung zu aktivieren. Hierbei werden mögliche Visualisierungen hinzugefügt (Multiplikationsaufgaben als Wendepflichtchen-Tabelle visualisiert, Additionsaufgaben als Zahlenstrahl und Tabellen visualisiert)
- In der Aufgabenerstellung kann man eigene Bilder hochladen. Diese kann man Zentrieren.
- In der Aufgabenerstellung kann man eigene Texte hinzufügen, bei welchen man die Absätze, Schriftarten und Schriftgrößen bestimmen kann.

<b>ID</b>	US-4
<b>Bezeichnung</b>	Erstellen von Vorlagen
<b>Beschreibung</b>	Als SHP möchte ich verschiedene Vorlagen erstellen können, welche für die Strukturierung der generierbaren Arbeitsblätter genutzt werden können. Über diese Vorlagen möchte ich steuern können, wie das Layout und die Gestaltung eines Aufgabenblatt aussehen sollte. Zudem habe ich gewisse Kinder, welche von weniger Aufgaben profitieren. Dies soll ebenfalls in einer Vorlage gespeichert werden können.

Tabelle 4: Functional Requirements (US-4)

**Akzeptanzkriterien (US-4)**

- Auf der Seite zum Erstellung von Vorlagen kann man die Abstände (Blöcke, Zeilen), die Schrift (Schriftart, Schriftgrösse, Farbe) für Titel und Text, die Anzahl Aufgaben pro Block und pro Seite konfigurieren.
- Die einzelnen Vorlagen können benannt werden.

<b>ID</b>	US-5
<b>Bezeichnung</b>	Verwalten von Vorlagen
<b>Beschreibung</b>	Als SHP möchte ich eine Übersicht über meine ganzen Vorlagen erhalten. Die einzelnen Vorlagen möchte ich löschen, umbenennen und bearbeiten können.

Tabelle 5: Functional Requirements (US-5)

**Akzeptanzkriterien (US-5)**

- Auf der Vorlagenübersicht findet man alle Vorlagen
- Die Vorlagen können gelöscht, umbenannt und bearbeitet werden.

<b>ID</b>	US-6
<b>Bezeichnung</b>	Generieren von Arbeitsblättern als PDF
<b>Beschreibung</b>	Als SHP möchte ich aus meinen bereits erstellten Aufgabensammlungen und Vorlagen ein Arbeitsblatt in Form eines PDF generieren können.

Tabelle 6: Functional Requirements (US-6)

**Akzeptanzkriterien (US-6)**

- Auf der Übersicht der Aufgabensammlungen kann man eine Aufgabensammlung auswählen für das Generieren eines PDF.
- Einer ausgewählten Aufgabensammlung kann man eine Vorlage hinzufügen.
- Die einzelnen Eigenschaften einer Vorlage können auf einer Konfigurationsseite verfeinert werden.
- Anhand der Aufgaben und der Vorlage wird ein PDF-Dokument zusammengestellt, welches in einer Browseransicht heruntergeladen werden kann.

- Die Parameter der Vorlage werden genutzt, um das Arbeitsblatt entsprechend zu Strukturieren und zu Gestalten.
- Jeder Aufgabentyp enthält eine entsprechende Visuelle Anzeige auf dem PDF-Dokument.
- Ein generiertes PDF enthält immer ein Namensfeld in der Kopfzeile
- Die einzelnen Aufgabenblöcke in einem generierten PDF werden durch farbliche Schattierung abgetrennt.

<b>ID</b>	US-7
<b>Bezeichnung</b>	Authentifizierung
<b>Beschreibung</b>	Als SHP möchte ich nicht, dass andere Personen auf meine Aufgabensammlungen und Vorlagen zugreifen können.

Tabelle 7: Functional Requirements (US-7)

### Akzeptanzkriterien (US-7)

- Es gibt ein Benutzerkonto für die SHP, bei dem sie sich mit Benutzernamen und Passwort anmelden kann.
- Im Rahmen des Prototypes, soll der Zugriff auf die Elemente in der Applikation nur durch ein valides Benutzerkonto möglich sein.

### 5.1.2 Secondary Features

<b>ID</b>	US-8
<b>Bezeichnung</b>	Typen von Aufgaben 2
<b>Beschreibung</b>	Als SHP möchte ich verschiedene Aufgabentypen der Mathematik zu einer Aufgabensammlung hinzufügen können. Ich möchte mich nicht darum kümmern einzelne aufgaben auszufüllen, sondern möchte, dass das System mir beliebig viele Aufgaben generiert. Da ich Schüler mit Lernschwächen betreue möchte ich für gewisse Aufgaben sinnvolle Visualisierungen hinzufügen können, um die Schüler zu unterstützen.

Tabelle 8: Functional Requirements (US-8)

**Akzeptanzkriterien (US-8)** In der Aufgabenerstellung kann man folgende Aufgabentypen auswählen:

- Bruchaufgaben (Mit Visualisierung eines Kuchendiagramms)
- Gleichungen
- Währungsaufgaben

<b>ID</b>	US-9
<b>Bezeichnung</b>	Übersetzungen
<b>Beschreibung</b>	Als SHP habe ich gewisse Kinder, welche Deutsch als Zweitsprache haben. Für diese Kinder möchte ich jeweils den deutschen Text in ihre Muttersprache übersetzen können.

Tabelle 9: Functional Requirements (US-9)

**Akzeptanzkriterien (US-9)**

- Übersetzungsbedürfnisse können einer Vorlage hinzugefügt werden.
- Beim Generieren eines PDF werden die Texte in die gewählte Sprache übersetzt und als Subtitel angezeigt.

<b>ID</b>	US-10
<b>Bezeichnung</b>	Silbenschrift
<b>Beschreibung</b>	Als SHP habe ich gewisse Kinder, welche die Aufgaben besser verstehen, wenn sie als Silbenschrift geschrieben werden. Ich möchte die Möglichkeit haben den ganzen Text in einem Arbeitsblatt, als Silbenschrift darzustellen.

Tabelle 10: Functional Requirements (US-10)

**Akzeptanzkriterien (US-10)**

- In einer Vorlage kann man einstellen, ob ein PDF Dokument die Texte als Silbenschrift darstellen soll.
- Beim Generieren eines PDF werden die Texte als Silbenschrift dargestellt.

## 5.2 Nicht-funktionale Anforderungen

**Kategorien** Als Vorlage für die nicht funktionalen Anforderungen werden die acht Kategorien aus dem ISO 25010 Standard[19] verwendet (Siehe Abbildung 31: ISO 25010 Qualitätscharakteristiken).

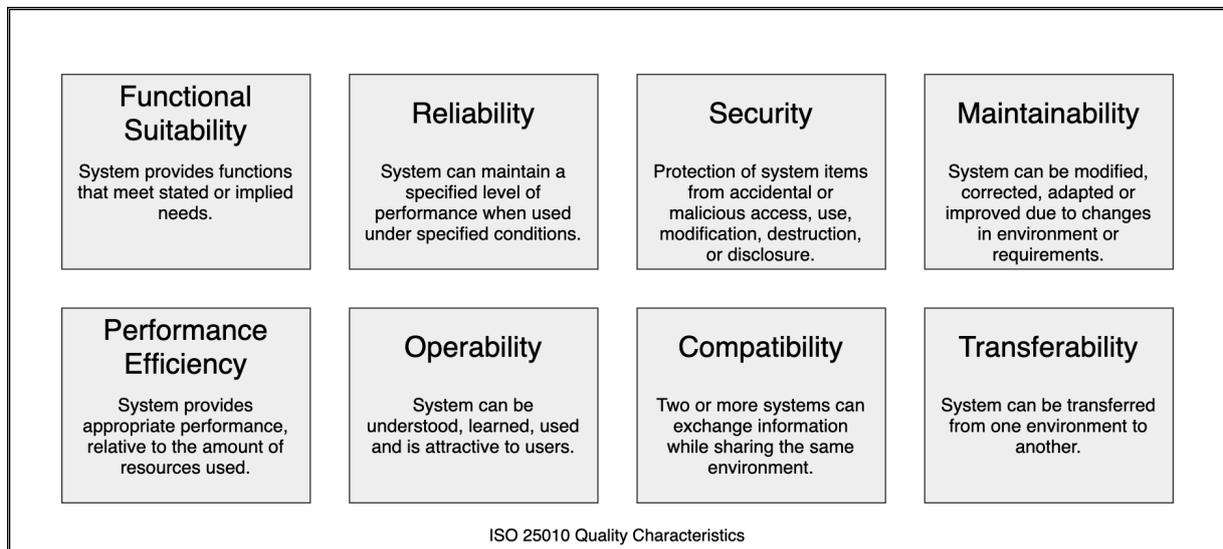


Abbildung 31: ISO 25010 Qualitätscharakteristiken

Quelle: <https://docs.arc42.org/images/1-2-iso-25010-topics-en.png>

**Kontrolle der Anforderungen** Jede Anforderung enthält Schritte welche definieren wie und durch wen die Erfüllung der Kriterien sichergestellt wird. Um diese Schritte sinnvoll wählen zu können verwenden wir die SMART[30] Prinzipien damit Anforderungen eindeutig messbar und prüfbar formuliert sind.

- **S:** Spezifisch
- **M:** Messbar
- **A:** Erreichbar (Achievable)
- **R:** Angemessen (Reasonable)
- **T:** Terminiert

Spezifisch und Messbar sind die wichtigsten Kriterien und werden jeweils pro Anforderung unter 'Beschreibung' definiert. Die anderen sind per Definition schon erfüllt.

- **Erreichbar & Angemessen:** Anforderungen wurden so gewählt, dass sie im Umfang der Arbeit erreichbar und angemessen sind. Es wurde Wert darauf gelegt einen funktionalen Prototypen zu erstellen, welcher in sich geschlossen verwendet werden kann.
- **Terminiert:** Die Arbeit hat ein definiertes Ende, die Anforderungen sollen bis Ende Abgabe erfüllt sein.

**Wartbarkeit (Maintainability)**

<b>ID</b>	NFR-1
<b>Bezeichnung</b>	Erweiterbarkeit
<b>Beschreibung</b>	Die Codebasis muss modular und erweiterbar gestaltet sein, um zukünftige Anpassungen und Weiterentwicklungen ohne grossen Aufwand zu ermöglichen.
<b>Priorität</b>	Mittel
<b>Massnahmen</b>	Die Einhaltung der SOLID-Prinzipien stellt sicher, dass die Codebasis modular und wartbar bleibt. Zusätzlich wird darauf geachtet, dass neue Module unabhängig hinzugefügt oder verändert werden können, ohne bestehende Funktionalität zu beeinträchtigen.
<b>Erfüllungskriterien</b>	Der Reviewer überprüft, ob die SOLID-Prinzipien in den relevanten Bereichen der Codebasis umgesetzt wurden.

Tabelle 11: Non Functional Requirements (NFR-1)

**Effizienz (Performance Efficiency)**

<b>ID</b>	NFR-2
<b>Bezeichnung</b>	Generierung Dokument
<b>Beschreibung</b>	Die Dauer zur Generierung eines druckfähigen Arbeitsblatts aus der HTML-Ansicht soll maximal 5 Sekunden betragen.
<b>Priorität</b>	Mittel
<b>Massnahmen</b>	Die Generierung erfolgt serverseitig und verwendet eine optimierte Bibliothek zur PDF-Erstellung. Recherche hilft geeignete Bibliothek zu finden.
<b>Erfüllungskriterien</b>	Die Performance wird während manuellen Tests gemessen, wobei mindestens 10 Arbeitsblätter mit unterschiedlichen Inhalten generiert werden. 90% der Tests dürfen die Dauer von 5 Sekunden nicht überschreiten.

Tabelle 12: Non Functional Requirements (NFR-2)

**Zuverlässigkeit (Reliability)**

<b>ID</b>	NFR-3
<b>Bezeichnung</b>	Input Validierung
<b>Beschreibung</b>	Benutzer sollen unmittelbar nach der Eingabe über fehlerhafte Eingaben informiert werden, um die Integrität der Daten zu gewährleisten und Fehler frühzeitig zu beheben.
<b>Priorität</b>	Hoch
<b>Massnahmen</b>	Alle Eingabefelder überprüfen, ob die Benutzereingaben den vordefinierten Validierungsregeln entsprechen. Fehlermeldungen müssen klar, präzise und kontextabhängig sein und den Benutzer darauf hinweisen, wie der Fehler zu beheben ist.
<b>Erfüllungskriterien</b>	Die Input-Validierung wird durch manuelle Tests überprüft.

Tabelle 13: Non Functional Requirements (NFR-3)

<b>ID</b>	NFR-4
<b>Bezeichnung</b>	Asynchrone Handlungen
<b>Beschreibung</b>	Die Benutzeroberfläche darf bei längeren Operationen nicht blockieren. Aktionen wie das Laden von Daten oder das Speichern von Informationen werden asynchron im Hintergrund durchgeführt, um die Benutzererfahrung nicht zu beeinträchtigen.
<b>Priorität</b>	Hoch
<b>Massnahmen</b>	Asynchrone Prozesse, wie Datenabfragen und das Speichern von Arbeitsblättern, werden mit asynchronen Aufrufen implementiert. Die UI soll währenddessen eine visuelle Rückmeldung in Form von Ladeindikatoren oder Fortschrittsbalken anzeigen, um den Benutzer über den laufenden Prozess zu informieren.
<b>Erfüllungskriterien</b>	Manuelle Tests stellen sicher, dass Ladeindikatoren korrekt angezeigt werden und dass nach Abschluss der asynchronen Aktionen das Ergebnis korrekt verarbeitet wird.

Tabelle 14: Non Functional Requirements (NFR-4)

**Betriebsfähigkeit (Operability)**

<b>ID</b>	NFR-5
<b>Bezeichnung</b>	Schulungsbedarf
<b>Beschreibung</b>	Die Benutzeroberfläche der Applikation muss so gestaltet sein, dass ein Benutzer nach maximal 20 Minuten Schulung die grundlegenden Funktionen der Applikation sicher verwenden kann.
<b>Priorität</b>	Hoch
<b>Massnahmen</b>	Die Benutzeroberfläche folgt etablierten Designprinzipien nach Norman/Nielsen Usability Heuristiken.
<b>Erfüllungskriterien</b>	Usability-Tests mit mindestens 2 Benutzern werden durchgeführt, um zu überprüfen, ob die Anforderung erreicht wird. In den Tests wird beobachtet, wie die Benutzer ohne Instruktionen mit der Applikation umgehen und Unverständlichkeiten werden dokumentiert.

Tabelle 15: Non Functional Requirements (NFR-5)

**Übertragbarkeit (Transferability)**

<b>ID</b>	NFR-6
<b>Bezeichnung</b>	Plattformunabhängigkeit
<b>Beschreibung</b>	Die Applikation soll einfach auf verschiedene Systeme übertragbar sein, mit minimaler Konfiguration und ohne Änderungen am Quellcode.
<b>Priorität</b>	Hoch
<b>Massnahmen</b>	Docker-Container und Docker Compose werden verwendet, um die Applikation und ihre Abhängigkeiten zu kapseln.
<b>Erfüllungskriterien</b>	Manuelle Tests stellen sicher, dass bei Portierung von bestehenden Applikationen nur der persistierte Ordner der Datenbank verschoben werden muss.

Tabelle 16: Non Functional Requirements (NFR-6)

**Sicherheit (Security)**

<b>ID</b>	NFR-7
<b>Bezeichnung</b>	Sichere Datenübertragung
<b>Beschreibung</b>	Alle Daten, die zwischen dem Browser und den Docker-Containern übertragen werden, müssen verschlüsselt sein, um die Vertraulichkeit und Integrität der Daten zu gewährleisten.
<b>Priorität</b>	Hoch
<b>Massnahmen</b>	Alle API-Kommunikationen müssen über HTTPS erfolgen. Eine unsichere HTTP-Verbindung kann nicht etabliert werden.
<b>Erfüllungskriterien</b>	Die sichere Datenübertragung wird durch manuelle Überprüfungen sichergestellt.

Tabelle 17: Non Functional Requirements (NFR-7)

<b>ID</b>	NFR-8
<b>Bezeichnung</b>	Authentifizierung
<b>Beschreibung</b>	Zugriff zu den Arbeitsblättern soll nur von authentifizierten Benutzern möglich sein.
<b>Priorität</b>	Hoch
<b>Massnahmen</b>	Einsatz des ASP.NET Core Identity Framework zur Authentifizierung der Benutzer.
<b>Erfüllungskriterien</b>	Authentifizierung vor der Verwendung der Applikation wird manuell sichergestellt.

Tabelle 18: Non Functional Requirements (NFR-8)

## 6 Technologien

Für die Implementierung des Prototypen wurden verschiedene Technologien eingesetzt. Dabei wurde auf die Stärken und Schwächen der Teammitglieder geachtet, um eine möglichst hohe Produktivität bereitzustellen. Zudem wurde darauf geachtet, zukunftssichere Technologien auszuwählen, welche untereinander kompatibel einsetzbar sind.

### 6.1 Blazor

Als Frontend-Framework wurde Blazor[4] gewählt. Blazor ist ein von Microsoft entwickeltes Framework, welches ermöglicht, interaktive Webanwendungen mit C# anstelle von JavaScript zu erstellen. Hier konnte auf bestehende Erfahrungen mit Blazor aus früheren Projekten zurückgegriffen werden, wodurch der Einarbeitungsaufwand reduziert wurde.

Ein wesentlicher Vorteil von Blazor ist die nahtlose Integration in das .NET-Ökosystem, welches eine Vielzahl von Bibliotheken und Framework anbietet und so den Entwicklungsprozess erleichtern. Auch die einfache Datenanbindung, sowie die Möglichkeit, dynamische und interaktive Benutzeroberflächen zu gestalten, machen Blazor zu einer geeigneten Wahl. Da dieses Projekt als prototypische Implementierung umgesetzt wird, war zudem die Zukunftssicherheit des gewählten Framework von Bedeutung. Blazor, als Teil der Microsoft .NET-Strategie, wird kontinuierlich weiterentwickelt und verbessert, was den Anforderungen an eine nachhaltige Technologie entspricht.

Blazor bietet unterschiedliche Rendermodi, welche eine Ausführung der Anwendung entweder auf dem Server- oder im WebAssembly-Modus ermöglichen. Für dieses Projekt wurde die WebAssembly-Variante von Blazor gewählt, da sie den C#-Code clientseitig im Browser ausführt. Dies reduziert die Serverlast, da weniger Berechnungen serverseitig durchgeführt werden müssen. Darüber hinaus ermöglicht dieser Ansatz eine native, app-ähnliche Benutzererfahrung, da die Anwendung im Browser läuft, ohne dass sie ständig neu geladen werden muss.

### 6.2 MudBlazor

Für die Gestaltung der Benutzeroberfläche wurde die Bibliothek MudBlazor[25] verwendet. MudBlazor zeichnet sich durch seine einfache Implementierung und die umfangreichen Konfigurationsmöglichkeiten aus, welche speziell für dieses Projekt eine effiziente Gestaltung der Benutzeroberfläche ermöglichten.

### 6.3 ASP.NET Core Web API

Für die Implementation der API-Schnittstelle zur Datenbank wurde ein ASP.NET Core Web API Projekt ausgewählt. Die ASP.NET Core Web API lässt sich nicht nur einfach mit Blazor integrieren, sondern unterstützt auch die leichte Integration mit anderen Frontend Frameworks. Allgemein ist eine ASP.NET Core Web API sehr performant und benötigt weniger Ressourcen und kann zudem auf verschiedenen Betriebssystemen entwickelt und bereitgestellt werden. Der ausschlaggebende Punkt für die Wahl ist jedoch die moderne Authentifizierungs- und Autorisierung-Funktionalität, welche bereits integriert ist. Sie ermöglicht die vereinfachte Implementierung von Authentifizierungs- und Autorisierungsmechanismen.

### 6.4 Entity Framework Core

Entity Framework Core ist in der Welt der ASP.NET Core-Technologien eine weit verbreitete Wahl zur Anbindung einer Datenbank. EF Core ist ein ORM-Framework und ermöglicht es

Entwicklern, mit Datenbanken zu interagieren, indem sie C#-Objekte verwenden. Dies reduziert die Notwendigkeit, SQL-Abfragen manuell zu schreiben, und ermöglicht eine einfachere Interaktion mit Daten. Entity Framework Core kann mit einer Vielzahl von Datenbankanbieter verwendet werden und ist ebenfalls Plattformunabhängig. Im Rahmen dieses Projektes, wurde Entity Framework Core vor allem gewählt, da es Datenbank abfragen mit LINQ ermöglicht. LINQ ist eine Funktion in .NET, die es ermöglicht, Abfragen direkt in C# zu formulieren. Mit LINQ kann man zum Beispiel Abfragen wie:

**”Gib mir alle Produkte, welche einen Preis von über 20 Franken haben.”**

Einfach darstellen als:

```
products.Where(p => p.Price > 20);
```

Listing 1: Anwendungsbeispiel: LINQ

## 6.5 Andere Frameworks & Packages

Neben den Hauptkomponenten wurden noch weitere Technologien verwendet, welche den Implementierungsprozess unterstützt haben.

**ASP.NET Identity API** Die Applikation ist aus dem Internet erreichbar, weswegen der Zugriff gegen Missbrauch geschützt sein muss. Aus diesem Grund wurde eine Nicht-Funktionale Anforderung (Siehe Kapitel 5.2 Abschnitt: Kontrolle der Anforderungen) definiert, die sicherstellt, dass nur authentifizierte und autorisierte Benutzer Zugriff erhalten. Microsoft bietet mit ASP.NET Core Identity ein Framework, welches speziell für solche Anforderungen entwickelt wurde. Es übernimmt für den aktuellen Prototypen die Verwaltung von Benutzern, deren E-Mail-Adressen und Passwörtern, und kann in Zukunft erweitert werden, um eine vollumfängliche Authentifizierung und Autorisierung für verschiedene Benutzergruppen wie Administratoren, Lehrer und Schüler zu ermöglichen.

Ein weiterer Vorteil von ASP.NET Core Identity ist die Möglichkeit, die Endpunkte der API abzusichern und den Zugriff auf bestimmte Seiten im Frontend zu beschränken, so dass sie nur berechtigte Benutzer einsehen können. Die Verbindung zur Datenbank, sowie die Erstellung der erforderlichen Tabellen, sind bereits in das Framework integriert, was die Implementierung erheblich vereinfacht. Auch die automatisierte Erstellung von Identity-API-Endpunkten (z.B. /login) ist enthalten, was ermöglicht, Login-, Registrierungs- oder Logout-Funktionen direkt aus dem Frontend aus zu steuern.

Ein grosser Vorteil von ASP.NET Core Identity ist seine hohe Erweiterbarkeit. Dies ist für den Prototypen von hoher Bedeutung und erlaubt die Weiterentwicklung des Produkts. Das Framework bietet umfassende Möglichkeiten zur Anpassung und Erweiterung, zum Beispiel durch spezielle Eigenschaften für Benutzer, die Verwendung von Benutzerforderungen und Rollen zur Steuerung des Zugriffs sowie die Integration von OAuth2- und OpenID-Connect-Protokollen. Weiter können auch benutzerdefinierte Authentifizierungsprozesse implementiert werden. Beispielsweise könnten Multi-Faktor-Authentifizierung (MFA) oder die Integration von Drittanbieter-Identitätsdiensten, hinzugefügt werden. Das macht ASP.NET Core Identity zu einer flexiblen Lösung, die mit den Anforderungen der Anwendung skaliert und sich den Bedürfnissen anpassen kann.

**Docker** Docker[9] ist eine Open-Source-Plattform für das dockerisieren von Anwendungen. Mit Docker können Entwickler ihre Software zusammen mit allen benötigten Abhängigkeiten

(wie Bibliotheken, Konfigurationsdateien und Laufzeitumgebungen) in standardisierte Container verpacken. Diese Container sind in sich abgeschlossene, leichtgewichtige Einheiten, die überall konsistent ausgeführt werden können – sei es auf dem Entwicklungsrechner, in der Cloud oder auf physischen Servern. So kann Plattform-abhängiges Verhalten eliminiert werden, da die Ausführungsumgebung immer identisch ist.

Im entwickelten Prototyp wurde eine moderne Container-basierte Architektur implementiert, bei der die verschiedenen Komponenten in separate Docker Container aufgeteilt wurden. Diese Microservice-Architektur besteht aus drei primären Container: einen für die API, einem für das Frontend und einem für die Datenbank.

Die Aufteilung in die verschiedenen Container, ermöglicht eine besser Wartbarkeit der Anwendung. Jeder Container kann unabhängig voneinander entwickelt, getestet und bereitgestellt werden.

**FluentValidation** FluentValidation[15] ist eine beliebte Validierungsbibliothek für .NET-Anwendungen. Sie ermöglicht Entwicklern, komplexe Validierungslogik auf elegante und lesbare Weise zu implementieren. Die Validierungslogik wird dabei separat von den Modellklassen geführt, was zu einem besser wartbaren Code führt.

**PostgreSQL** Als Datenbanktechnologie wurde PostgreSQL ausgewählt. PostgreSQL ist ein zuverlässiges und modernes Datenbankmanagementsystem, welches sich sowohl für relationale als auch hybride Anwendungsfälle eignet. Die Entscheidung stand zunächst zwischen SQL Express und PostgreSQL. Dabei ist aufgefallen, dass PostgreSQL wesentlich performanter ist, vor allem bei parallelen Abfragen und der Verarbeitung von grossen Datenmengen, und sich somit besser für die Umsetzung dieses Projektes eignet.

## 7 Konzept und Architektur

Dieses Kapitel dient dem Verständnis der konzeptionellen Aspekte, welche für die erfolgreiche Umsetzung der Applikation aufgebaut wurden. Es soll die entworfene Systemarchitektur der Applikation beschreiben und visuell aufzeigen, wie die einzelnen Teilsysteme zusammenhängen. Dazu werden die vier Stufen des C4-Modells vorgestellt, welche ein umfassendes Verständnis der grundlegenden Struktur und des Entwurfs repräsentieren sollen.

Im letzten Abschnitt wird der Entwurf für das Hosting und Deployment der Applikation visuell dargestellt.

### 7.1 C4 Modell

Das C4-Modell ist ein Modell zur Visualisierung von Softwarearchitektur. Es besteht aus vier Ebenen: Context, Container, Component und Code. Die Stärke vom C4-Modell liegt darin, dass es einfach und verständlich ist und sich gut für die Kommunikation zwischen Entwicklern und Stakeholdern eignet.

#### 7.1.1 Level 1: Context Diagram

Das Context Diagram (Siehe Abbildung 32: C4 Context Diagramm) soll aufzeigen, wie das System in die Umgebung eingebettet ist. Es zeigt die Interaktionen zwischen dem System und den externen Akteuren. Unter einem Akteur kann eine Person, ein anderes System oder eine Hardwarekomponente verstanden werden.

Bei der geplanten Webapplikation, können die Benutzer\*innen mit dem Software System interagieren.

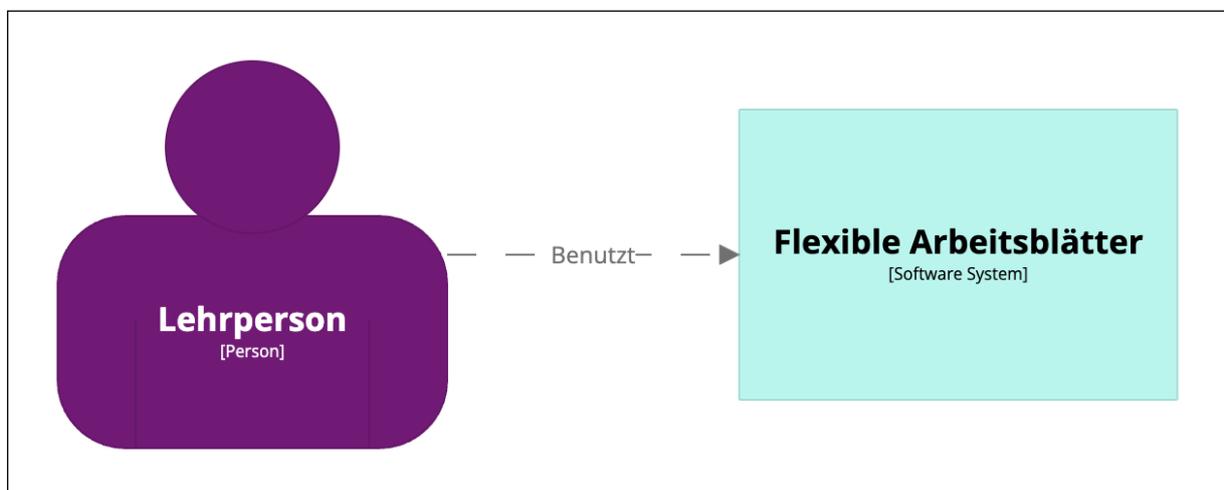


Abbildung 32: C4 Context Diagramm

Quelle: *Erstellt mit Structurizr [31]*

### 7.1.2 Level 2: Container Diagram

Das Container Diagramm (Siehe Abbildung 33: C4 Container Diagramm) zeigt die grobe Architektur des Softwaresystems, sowie die Interaktionen zwischen den Hauptkomponenten.

Die Benutzer\*innen interagieren mit dem Softwaresystem über eine Web-Applikation. Die Web-Applikation dient als Benutzeroberfläche und kommuniziert direkt mit der API-Schnittstelle. Die API übernimmt hierbei die Kommunikation zwischen der Web-Applikation und der Datenbank.

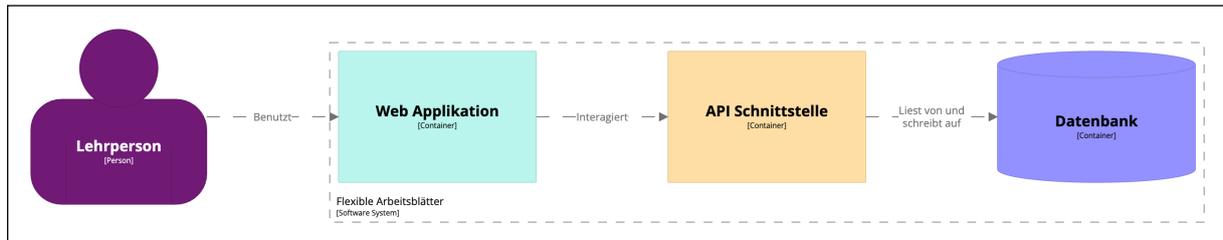


Abbildung 33: C4 Container Diagramm

Quelle: *Erstellt mit Structurizr [31]*

### 7.1.3 Level 3: Component Diagram

Das Component Diagram (Siehe Abbildung 34: C4 Component Diagramm) verfeinert die Architektur des Softwaresystems und zeigt die einzelnen Bausteine (Komponenten) sowie deren Interaktion im System.

In der Webapplikation gibt es Razor-Komponenten (Pages), welche zuständig sind für das Layout und die Gestaltung der einzelnen Seiten auf der Webapplikation. Die Pages haben Zugriff auf die clientseitige Businesslogik, mit welcher die gesamte Funktionalität der Applikation bereitgestellt wird. Über ein HTTP Repository Service wird sichergestellt, dass die Web Applikation auf die API Schnittstelle zugreifen kann und so die benötigten Daten vom Server laden kann.

In der API Schnittstelle (Server) existieren API Controller, welche die Daten im Data Context, über das Unit of Work-Pattern zugänglich machen sollen. Der Data Context ist schlussendlich die Kapselung zu den Entitäten welche auf der Datenbank liegen.

Da beide Komponenten (Web Applikation und API Schnittstelle) die gleichen Klassen (Models und Converter) benötigen, werden diese in ein Shared-Projekt ausgelagert. Auf dem Server, sowie auf dem Client, existiert Authentifizierungs Code, welcher für die Kommunikation mit dem ASP.NET Core Identity-Framework genutzt wird. Der Identity Controller wurde absichtlich grün eingefärbt, da dies generierter Code ist und nicht direkt zur Komponente gehört.

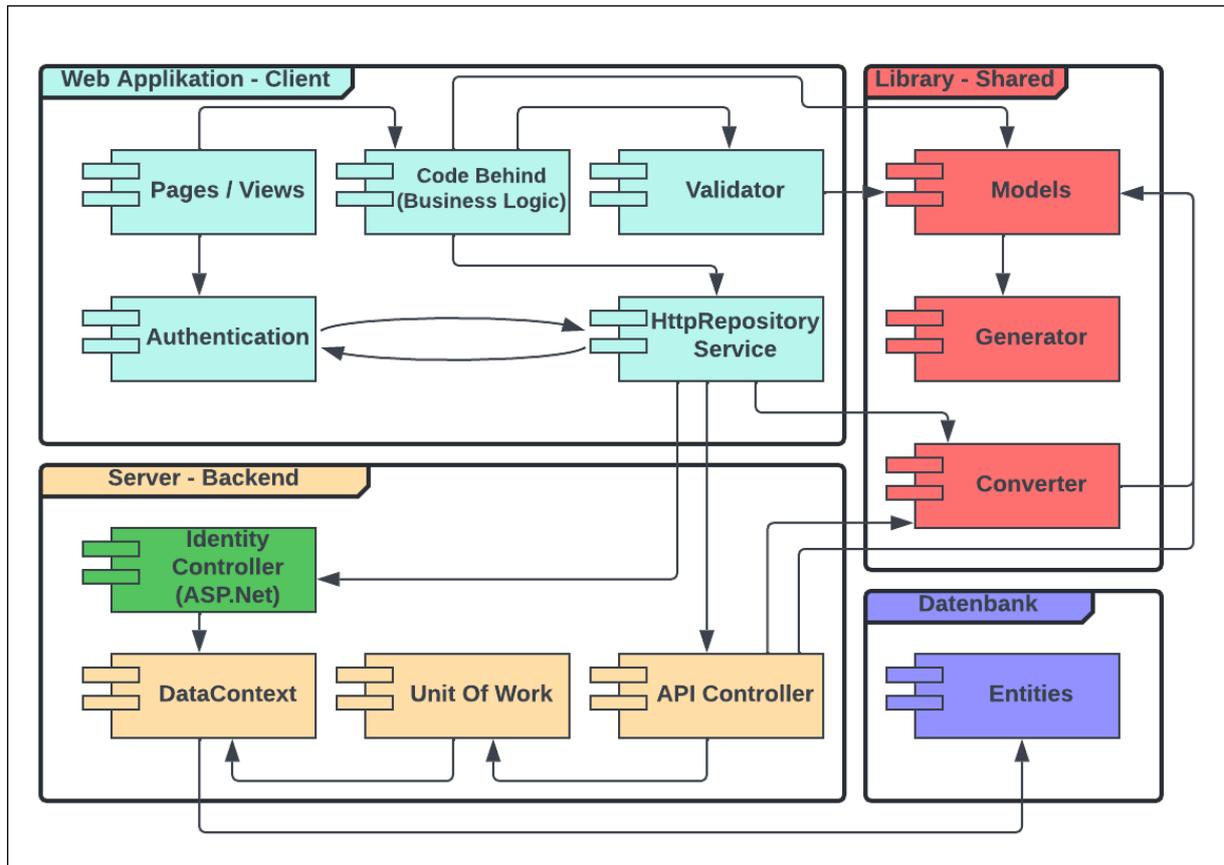


Abbildung 34: C4 Component Diagramm

Quelle: *Erstellt mit Lucidchart [22]*

### 7.1.4 Level 4: Class Diagram

Das Klassendiagramm zeigt die statische Struktur des Systems, indem es die wichtigsten Klassen, ihre Attribute, Methoden sowie die Beziehungen zwischen den Klassen darstellt.

**Client - Code-Behind (Business Logic)** Im Client befinden sich die Razor-Komponenten, welche ihre Businesslogik in einer Code-Behind-Klasse (Siehe Abbildung 35: Klassendiagramm: Business Logik der Pages) implementieren. Diese Klassen sind zuständig für jegliche Operationen, welche von den Benutzer\*innen getätigt werden können. Ein grosser Teil davon sind die CRUD-Operationen auf den Entitäten. Jede Code-Behind-Klasse (Ausser Login und Register) überschreibt dabei die `OnInitializedAsync()`-Methode in welcher eine einmalige Verbindung zum Server erstellt wird, um die entsprechenden Entitäten zu laden.

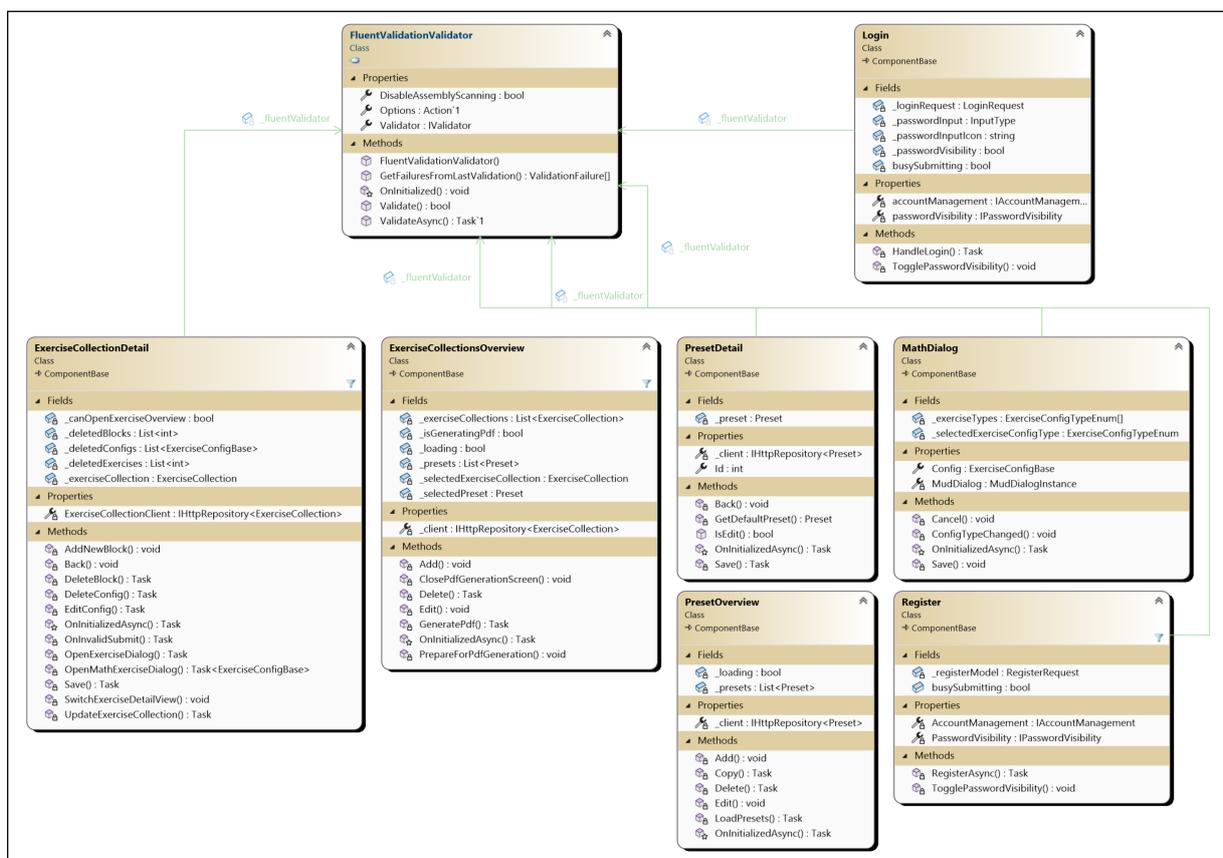


Abbildung 35: Klassendiagramm: Business Logik der Pages

Quelle: Erstellt mit VS Class Designer [35]

**Client - Validator** Damit die Benutzereingaben bereits clientseitig überprüft und den Benutzer\*innen als direktes Feedback angezeigt werden können, wurde der FluentValidationValidator verwendet. Dies ist ein Konzept aus der .NET-Welt und ermöglicht durch die FluentValidation-Bibliothek eine einfache Handhabung der Eingabevalidierung. Mit dieser Bibliothek können spezifische Validatoren (Siehe Abbildung 36: Klassendiagramm: Validatoren) für jegliche Model-Klassen erstellt werden. Die FluentValidation-Bibliothek bietet dazu die Möglichkeit eine Ableitung der Klasse `AbstractValidator<T>` zu erstellen und durch eine eigene und einfache Regelverwaltungs-API die Anforderungen an die Eigenschaften der Models zu erstellen.

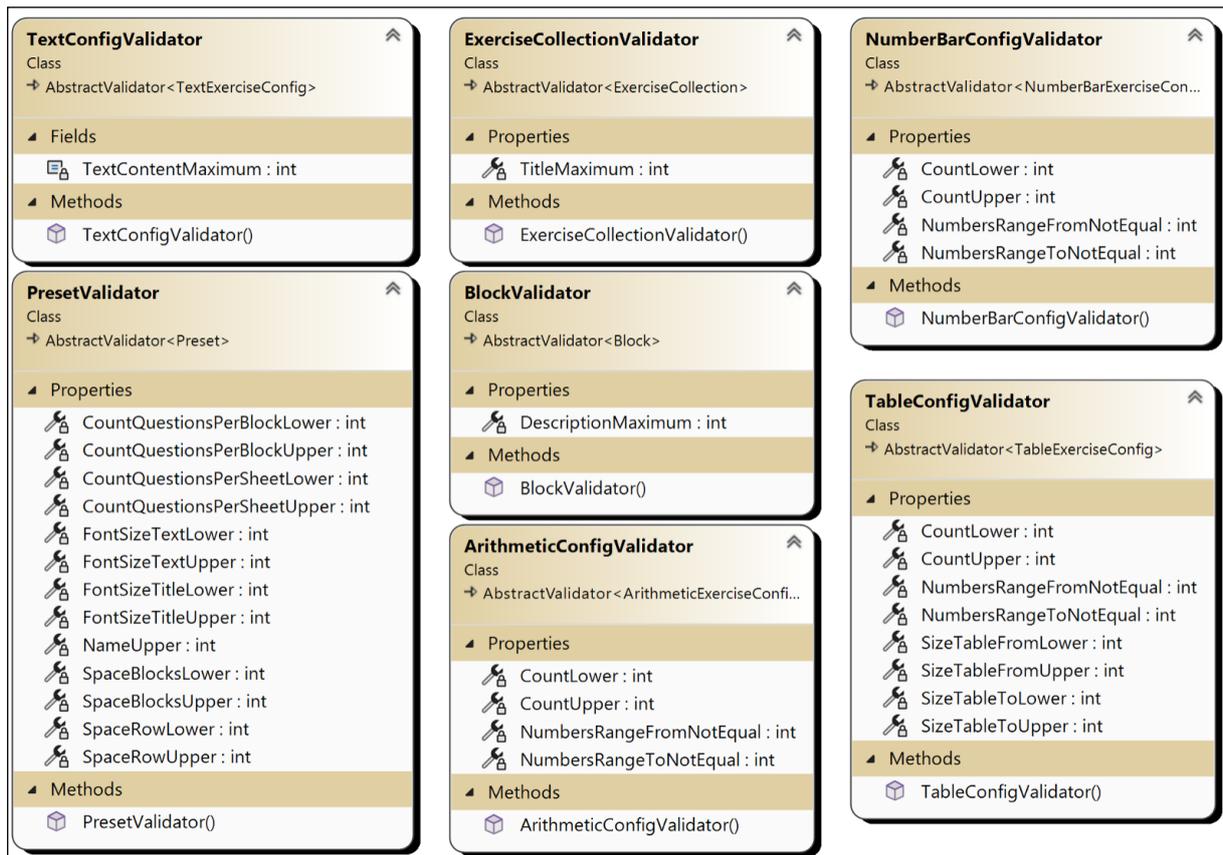


Abbildung 36: Klassendiagramm: Validatoren

Quelle: Erstellt mit VS Class Designer [35]

**Client - Http Repository** Die Kommunikation zum Server wird über ein strukturiertes HTTP Repository (Siehe Abbildung 37: Klassendiagramm: HTTP Repository) abgewickelt. Diese Architektur bietet eine klare Trennung zwischen der Geschäftslogik und der Kommunikationslogik mit dem Server. Das HTTP Repository definiert die grundlegenden Operation für die Interaktion mit dem Server. Diese Generischen Methoden ermöglichen es, dieselbe Schnittstelle für verschiedene Datentypen (T) zu verwenden, wodurch Wiederverwendbarkeit und Einheitlichkeit gewährleistet ist. Für die implementierten Methoden des `IHttpRepository`-Interface wird der integrierte `HttpClient` von .NET verwendet, welcher einfach ermöglicht die Daten über eine JSON Struktur zu senden und zu empfangen. Aufgrund der Verwendung von Polymorphie bei den Entitäten, müssen spezielle JSON-Konverter verwendet werden, um das Serialisieren und Deserialisieren richtig zu behandeln. Dafür wurde der `HttpClient` in eine Wrapper-Klasse gepackt und mit den Convertern (Siehe 8.1 Abschnitt "JSON Converter") ausgestattet.

Für die Zentrale Fehlerbehandlung aller HTTP-Anfragen wurde ein `HttpInterceptorService` von dem `Toolbelt.Blazor`-Paket genutzt. Dieser ermöglicht die Überwachung von HTTP-Anfragen und fängt fehlerhaft Serverantworten ab. So können spezifische Fehler entsprechend behandelt werden (z. B. Error 404 wird auf eine "Diese Seite existiert nicht"-Seite weitergeleitet). Im Rahmen dieses Prototyps wurden jedoch noch keine spezifischen Weiterleitungen und Fehlerhinweise implementiert. Die Fehler werden zur Zeit einfach abgefangen und man wird auf die Startseite weitergeleitet.

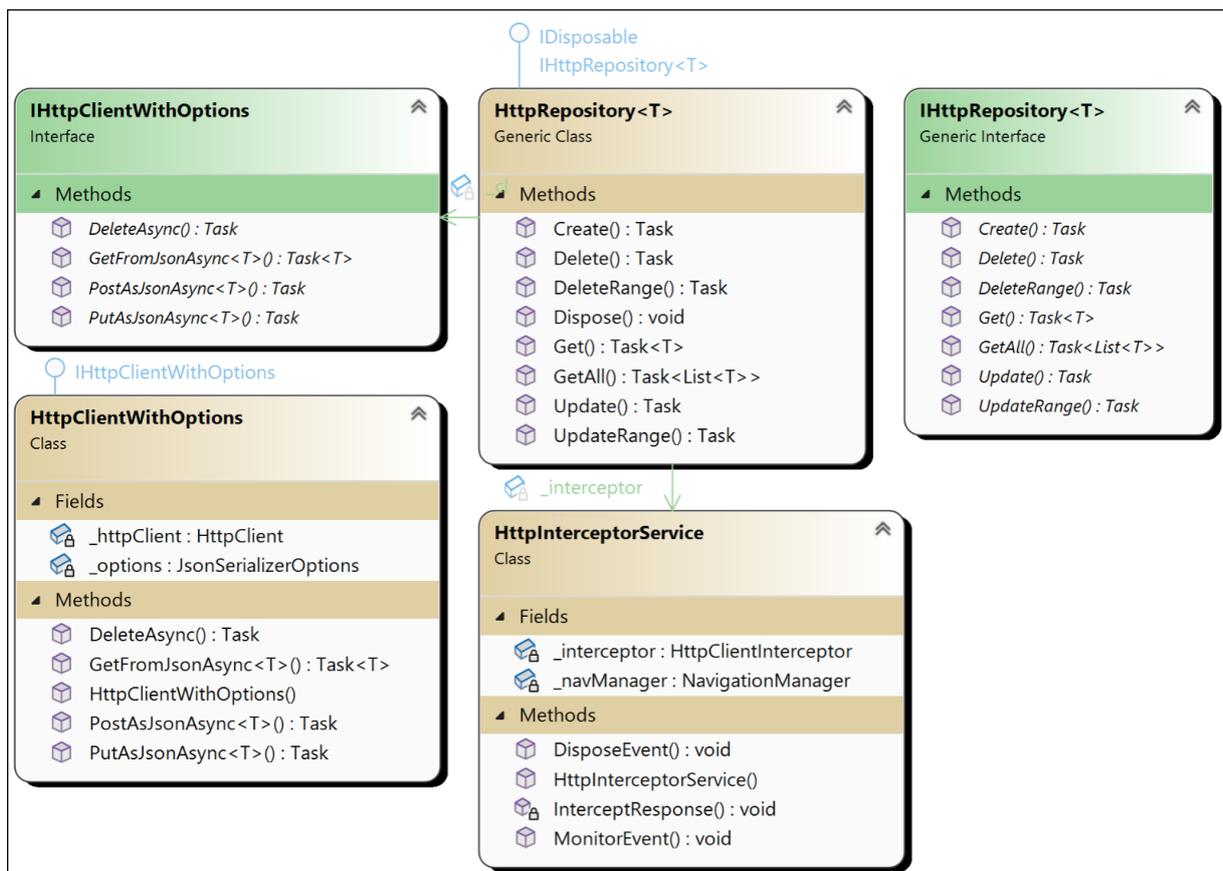


Abbildung 37: Klassendiagramm: HTTP Repository

Quelle: Erstellt mit VS Class Designer [35]

**Client - Authentication** Für den clientseitigen Authentifizierungsmechanismus (Siehe Abbildung 38: Klassendiagramm: Clientseitige Authentifizierung) werden mehrere Klassen benötigt.

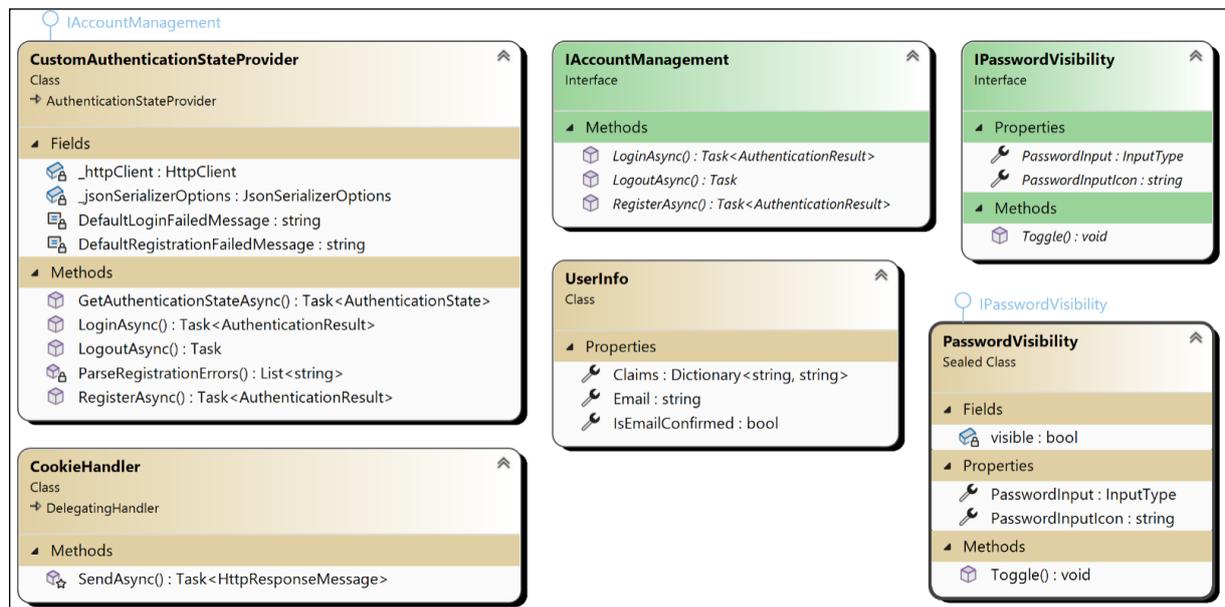


Abbildung 38: Klassendiagramm: Clientseitige Authentifizierung

Quelle: Erstellt mit VS Class Designer [35]

Die `CustomAuthenticationStateProvider` Klasse ist die zentrale Anlaufstelle für das Frontend wenn es um Authentifizierung geht. Sie kommuniziert mit den Endpunkte, welche von der ASP.NET Core Web API zur Verfügung gestellt werden. Folgende Abfragen werden von dieser Klasse abgewickelt:

- Abfragen des Authentifizierungs-Status an das Backend
- Registrieren eines neuen Benutzers
- Login eines bestehenden Benutzers
- Senden einer Logout Anfrage and den Server

Damit die Kommunikation mit den eigenen Endpunkte korrekt funktioniert, muss bei jeder HTTP-Anfrage ein Authentifizierungs-Cookie mitgeliefert werden. Diese Aufgabe wird vom `CookieHandler` übernommen

`PasswordVisibility` ist eine Klasse, welche auf der Benutzeroberfläche steuert, ob das Passwort in Plain-Text angezeigt wird oder nicht.

`UserInfo` dient als DTO zur Speicherung des Rückgabewertes vom `api/manage/info` Endpunkte, welcher überprüft ob ein Benutzer authentifiziert ist oder nicht.

Genauere Details zur Implementation finden sich im Kapitel 8.5: Authentifizierung.

**Server** Die Backend-Klassen (Siehe Abbildung 39: Klassendiagramm: Backend) verwenden eine Kombination aus dem Repository Pattern und dem Unit of Work-Pattern, um die Datenzugriffsschicht strukturiert und erweiterbar zu gestalten. Diese Patterns sorgen für eine klare Trennung zwischen der Geschäftslogik und der Datenbankinteraktion, wodurch der Code wartbarer und testbarer wird.

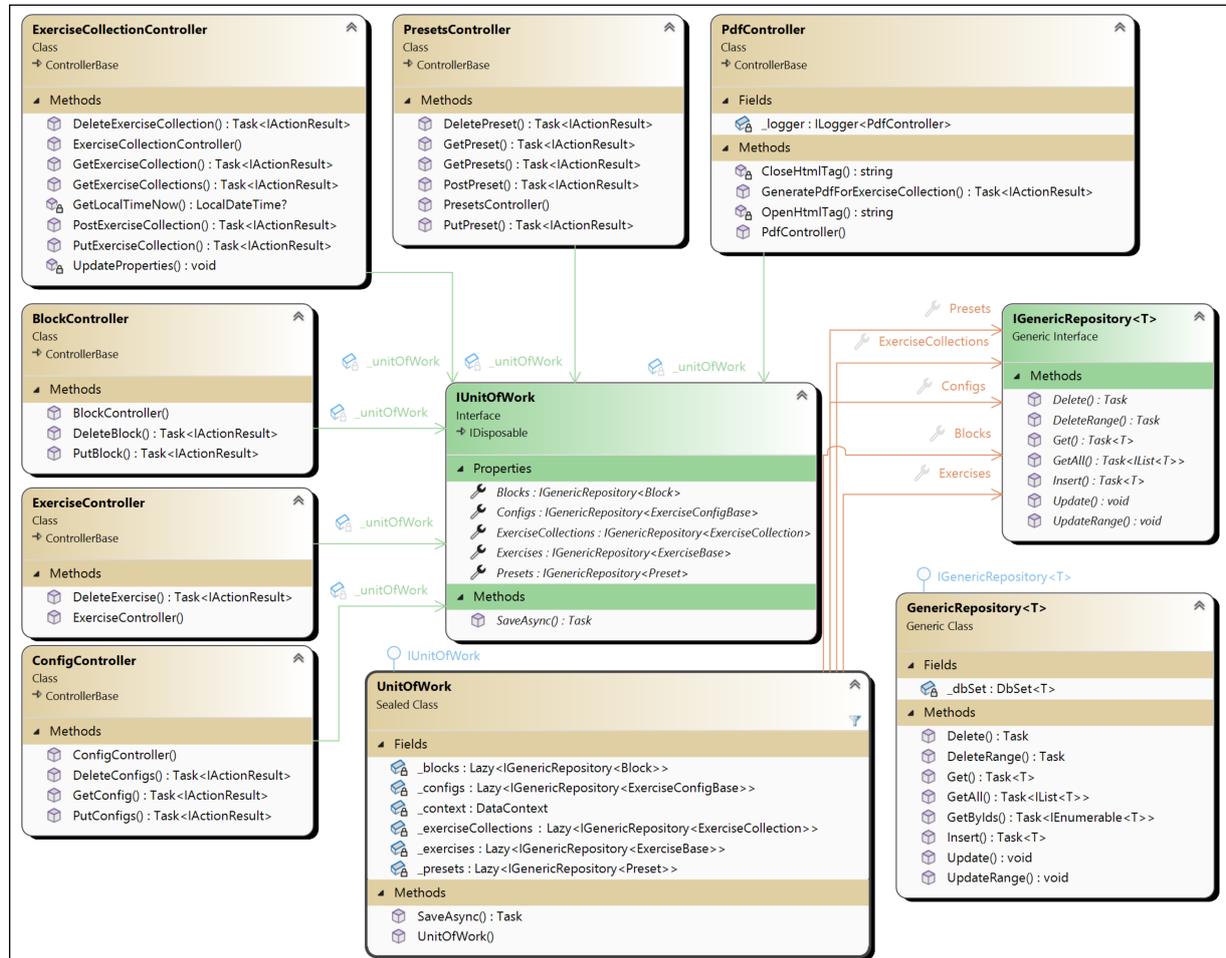


Abbildung 39: Klassendiagramm: Backend

Quelle: *Erstellt mit VS Class Designer [35]*

**Server - Repository Pattern** Das Repository Pattern abstrahiert die Datenbankzugriffe und bietet eine zentralisierte Schnittstelle, um Entitäten aus der Datenbank zu lesen, hinzuzufügen, zu aktualisieren oder zu löschen.

Durch die Parameter `filter`, `orderBy`, `includes` können einfache Abfragen mit C# erstellt werden, ohne dazu auf SQL zurückgreifen zu müssen: Das nachfolgende Beispiel holt alle Aufgabensammlungen, sortiert sie nach dem Erstellungsdatum und inkludiert alle verschachtelten Entitäten (Alle Abhängigkeiten zu anderen Tabellen)

```
var exerciseCollections = await _unitOfWork.ExerciseCollections.GetAll(
    orderBy: q => q.OrderBy(e => e.CreationDate),
    includes: q => q.Include(ec => ec.Blocks)
        .ThenInclude(b => b.Generators)
        .ThenInclude(g => g.Exercises))
```

Listing 2: Anwendungsbeispiel: Repository Pattern

Das Arbeiten mit so einer Struktur ist vor allem für einen .NET Entwickler wesentlich einfacher und verständlicher, als die daraus erzeugte SQL-Abfrage (Siehe Listing 3: SQL Alternative für 2: Anwendungsbeispiel: Repository Pattern).

```
1  SELECT ec.*,
2     b.*,
3     g.*,
4     e.*
5  FROM ExerciseCollections ec
6  LEFT JOIN Blocks b ON ec.Id = b.ExerciseCollectionId
7  LEFT JOIN Generators g ON b.Id = g.BlockId
8  LEFT JOIN Exercises e ON g.Id = e.GeneratorId
9  ORDER BY ec.CreationDate ASC;
```

Listing 3: SQL Alternative für 2: Anwendungsbeispiel: Repository Pattern

**Server - Unit Of Work Pattern** Das Unit of Work-Pattern koordiniert mehrere Repositories, um sicherzustellen, dass Datenbankoperationen konsistent und in einer einzelnen Transaktion (Einer einzelnen "Unit") ausgeführt werden.

Über die `SaveAsync`-Methode werden alle Änderungen gespeichert, welche über die Repositories gemacht wurden. Das Unit of Work-Pattern fasst alle Änderungen zusammen und speichert die Daten atomar.

**Server - ASP.NET Controllers** Die API-Controller bilden die zentrale Schnittstelle zwischen dem Client und dem Server im Backend. Sie sind zuständig für die Verarbeitung von HTTP-Anfragen. Die Controller wurden als RESTful-Services implementiert, welche die CRUD-Operation (Create, Read, Update, Delete) unterstützen. Alle Controller nutzen das Interface `IUnitOfWork`, um Datenoperation effizient zu kapseln.

**Shared - Models** Im Shared-Projekt liegen alle Models, welche vom Server und vom Client Projekt verwendet werden. Dabei ist zu beachten, dass ein Model in zwei Partial Klassen getrennt wurden, um die Funktionalitäten für das Frontend von den Eigenschaften, welche nachher auf der Datenbank liegen zu trennen. Partial Klassen, sind ein C#-Feature, welches die Übersichtlichkeit und Wartbarkeit des Codes, besonders bei grossen Klassen oder generiertem Code erheblich verbessert.

**Shared - Aufgabensammlung / Vorlage** Auf der höchsten Stufe der Models sind die Klassen für die Aufgabensammlungen und der Vorlagen (Siehe Abbildung 40: Klassendiagramm: Models (Aufgabensammlung / Vorlage)). Die Vorlagen-Klasse ist ein eigenständiges Model, welches keine weiteren Abhängigkeiten benötigt.

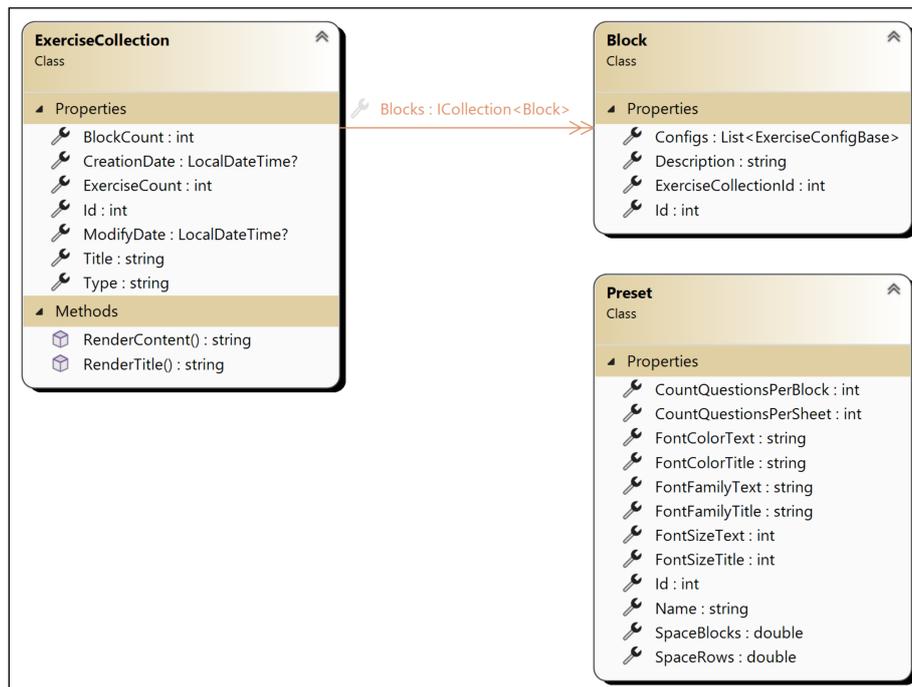


Abbildung 40: Klassendiagramm: Models (Aufgabensammlung / Vorlage)

Quelle: *Erstellt mit VS Class Designer [35]*

Die Aufgabensammlung enthält eine Liste von Funktionsblöcken, welche wiederum eine Liste von Aufgabenkonfigurationen enthalten.

**Shared - Aufgabenkonfigurationen** Da die Aufgabenkonfigurationen (Siehe Abbildung 41: Klassendiagramm: Models (Aufgabenkonfigurationen)) wiederverwendbar und editierbar sein sollen, müssen diese auch auf der Datenbank und entsprechend in den Models repräsentiert sein. Die Eigenschaften auf den Aufgabenkonfigurationen spiegeln direkt die einstellbaren Eigenschaften in den Dialogen der Aufgabenerstellung auf der Benutzeroberfläche wieder. Zudem enthalten die Aufgabenkonfigurationen die Möglichkeit die tatsächlichen Aufgaben zu generieren (Über den Aufgabengenerator (Siehe Abbildung 43: Klassendiagramm: Aufgabengenerator)) und die oberste Struktur für die Visualisierung der Aufgaben.

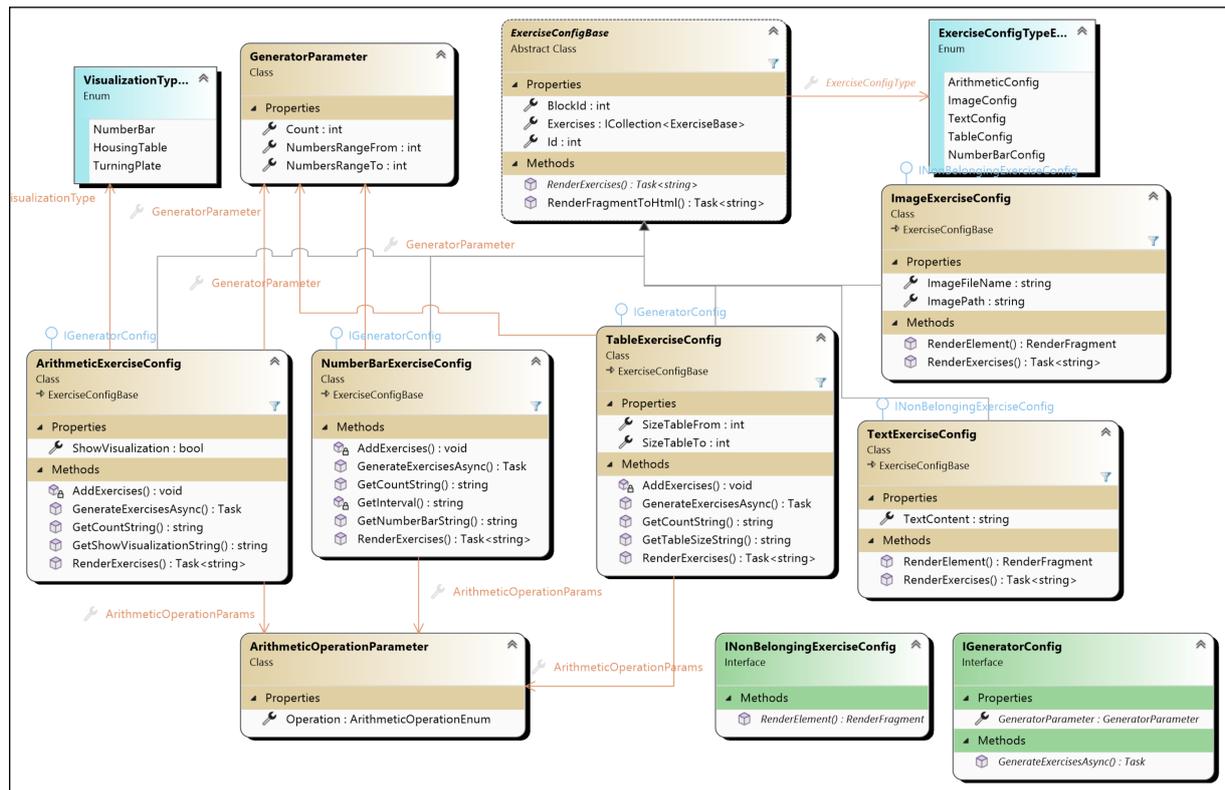


Abbildung 41: Klassendiagramm: Models (Aufgabenkonfigurationen)

Quelle: Erstellt mit VS Class Designer [35]

**Shared - Aufgaben** Für jede generierbare Aufgabe gibt es eine eigene Model-Klasse (Siehe Abbildung 42: Klassendiagramm: Models (Aufgaben)), welche die minimal benötigten Eigenschaften für das Speichern von Aufgaben enthält. Jedes Aufgabenmodell definiert dazu noch die Funktionalität für wie eine Aufgabe schlussendlich visualisiert werden soll.

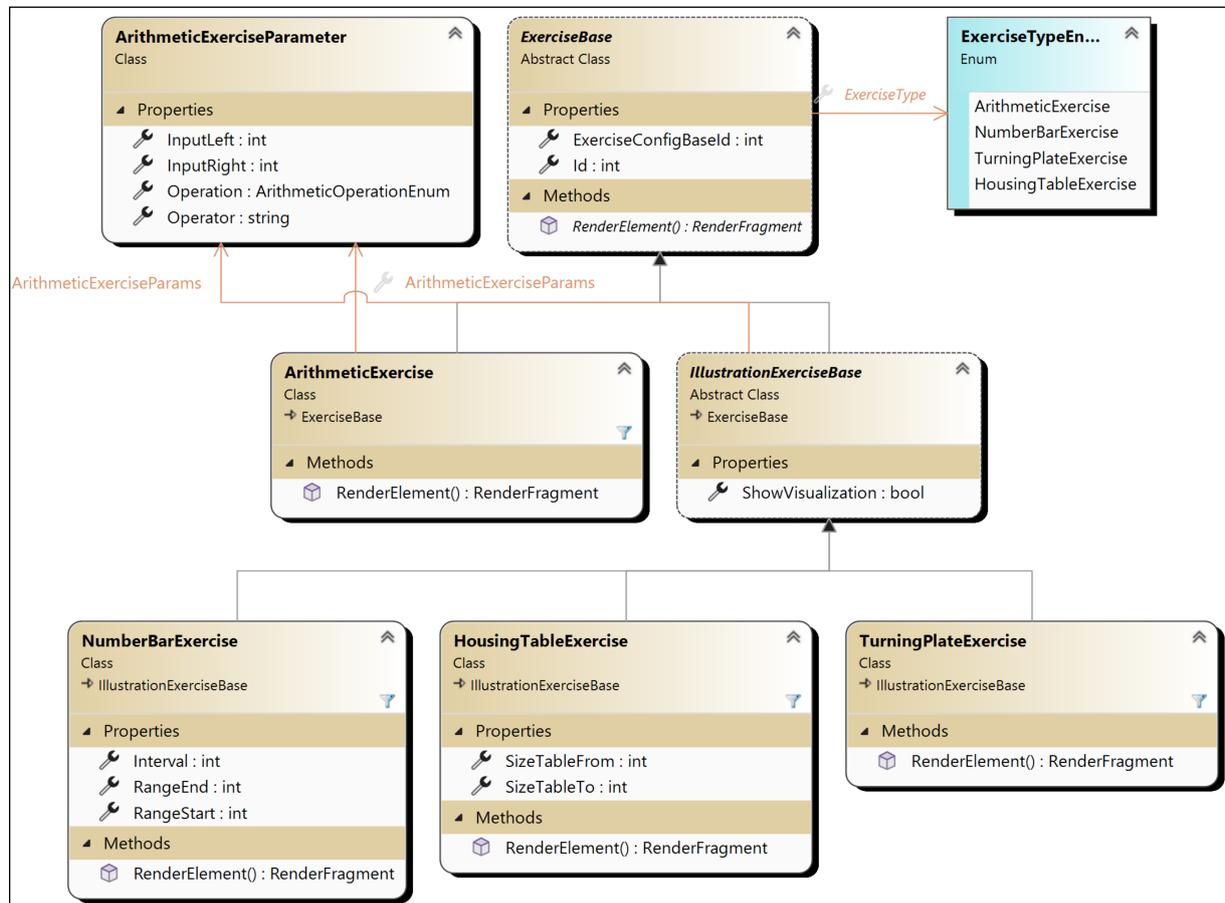


Abbildung 42: Klassendiagramm: Models (Aufgaben)

Quelle: Erstellt mit VS Class Designer [35]

**Shared - Generator** Über den Aufgabengenerator (Siehe Abbildung 43: Klassendiagramm: Aufgabengenerator) können die ganzen generierbaren Aufgaben (Alle ausser Text und Bilder, da diese nicht generierbar sind) erstellt werden. Für mehr Informationen, wie die Aufgaben genau generiert werden, siehe Kapitel 8.3: Generierung von Aufgaben.

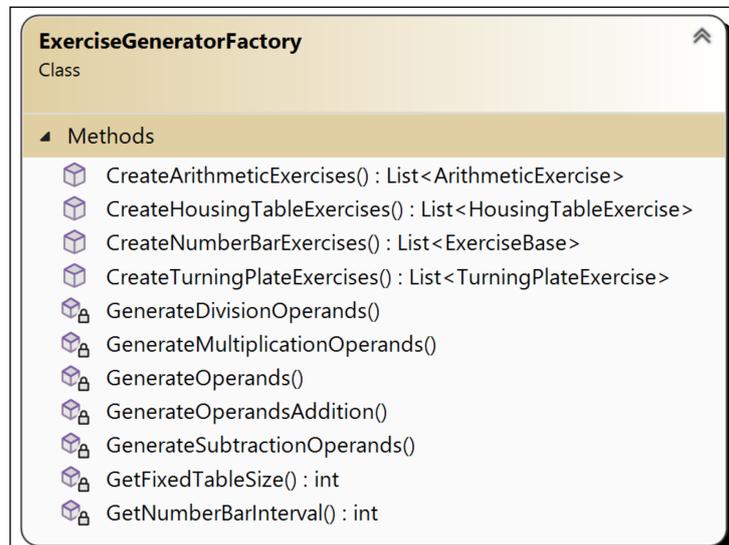


Abbildung 43: Klassendiagramm: Aufgabengenerator

Quelle: *Erstellt mit VS Class Designer [35]*

**Datenbank** Für das Erstellen der Datenbank-Tabellen wurde mit Entity Framework Core (EF Core) gearbeitet. EF Core ist ein leistungsstarkes ORM-Framework, welches ermöglicht die Datenbanktabellen aus C#-Klassen (Code-First Ansatz) oder C#-Klassen aus einer vorhandenen Datenbank (Database-First Ansatz) zu generieren.

**Code-First Ansatz** Für dieses Projekt wurde entschieden den Code-First Ansatz zu verwenden, da die Vertrautheit mit dem C#-Syntax ein grosser Vorteil war. Dabei wurden als erstes die Model-Klassen erstellt, welche die Struktur der Datenbank abbilden. Diese Klassen enthalten die Eigenschaften, die später zu den Spalten in den Datenbanktabellen werden.

Über einen Datenbankkontext (Siehe Abbildung 44: Klassendiagramm: Datenkontext), wird die Verbindung zur Datenbank verwaltet und die Modelle registriert.

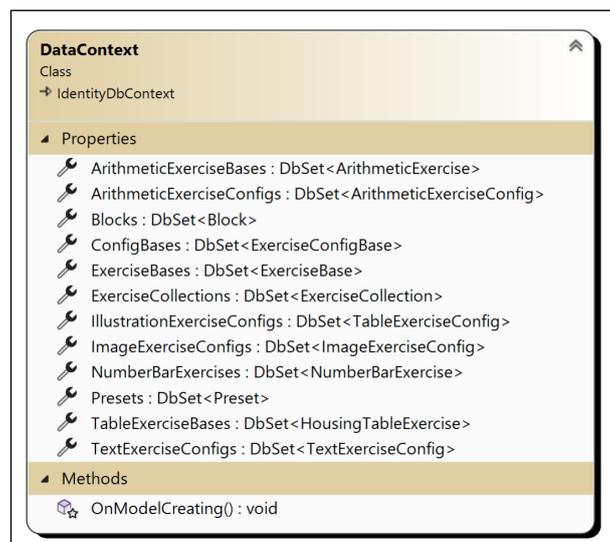


Abbildung 44: Klassendiagramm: Datenkontext

Quelle: *Erstellt mit VS Class Designer [35]*

**Mapping Strategien bei Polymorphismus** In den Models gibt es zwei polymorphe Strukturen. Einmal für die Aufgabenkonfigurationen und einmal für die Aufgaben an sich. Bei Entity Framework Core gibt es drei verschiedene Wege, wie man diese Strukturen in der Datenbank abbilden kann.

Die erste Strategie nennt sich "Table-Per-Hierarchy" (TPH). Hierbei werden alle Entitäten der Vererbungshierarchie in einer einzigen Tabelle gespeichert. Ein zusätzlicher Diskriminator muss verwendet werden, um zwischen den Typen zu unterscheiden.

#### Vorteile(+) / Nachteile (-)

- + Bessere Performance, da nur eine Tabelle abgefragt werden muss
- + Einfacher Verwaltung und geringerer Speicherplatzbedarf
- + Weniger Komplexität
- Nicht genutzte Spalten für abgeleitete Klassen (Null Werte)
- Kann bei einer sehr tiefen Hierarchie unübersichtlich werden

Die zweite Strategie nennt sich "Table-Per-Type" (TPT). Jede Klasse in der Vererbungshierarchie wird hierbei in einer separaten Tabelle gespeichert.

#### Vorteile(+) / Nachteile (-)

- + Spart Speicherplatz, da nur relevante Spalten in jeder Tabelle gespeichert werden

- Performanceprobleme bei Abfragen aufgrund der benötigten Joins
- Komplexeres Schema

Die letzte Strategie, welche erst seit EF Core 8.0 existiert, nennt sich "Table-Per-Concrete-Type" (TPC). Jede konkrete Klasse (Alle ausser Basis- / Abstrakten-klassen) hat hierbei ihre eigene Tabelle, welche alle geerbten Eigenschaften speichert.

### Vorteile(+) / Nachteile (-)

- + Keine Joins erforderlich
- + Einfacher zu verstehen, wenn nur konkrete Typen verwendet werden.
- Datenredundanz, da geerbte Eigenschaften in jeder Tabelle dupliziert werden.
- Änderungen an der Basisklasse können mehrere Tabellen beeinflussen

**Entscheid** Zum Start des Projektes, wurde die TPT-Mapping Strategie verwendet, da sie relativ simple zu verstehen ist und nicht viel Konfigurationsaufwand mit sich bringt. Nach einiger Zeit wurde aber bemerkt, dass die Performance dieser Strategie ungenügend ist. Die Ladezeiten waren zwar nicht schlimm, doch wiesen schon bei kleinen Entitätsmengen Mängel auf, weshalb entschieden wurde auf die TPH-Mappingstrategie umzusteigen, um auch bei grossen Datenmengen nicht zu lange Wartezeiten auf dem Client mit sich zu ziehen.

Entity Framework Core generiert schlussendlich die Datenbanktabellen (Siehe Abbildung 45 Datenbankdiagramm: Tabellen aus EF Core generiert) aus den Models.

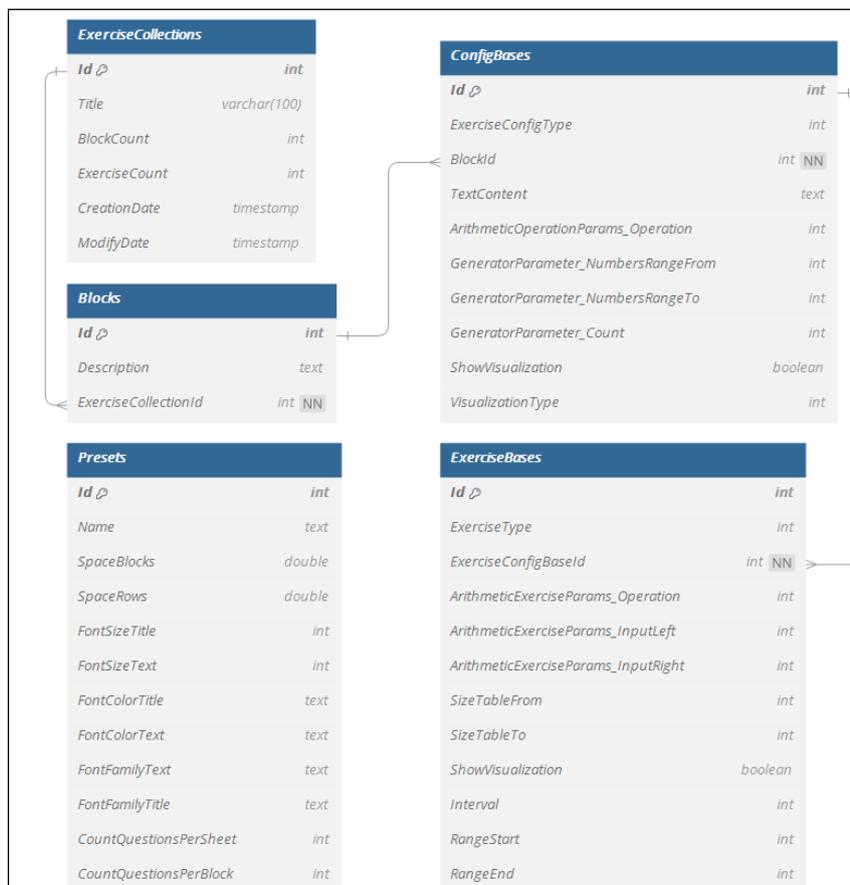


Abbildung 45: Datenbankdiagramm: Tabellen aus EF Core generiert

Quelle: *Erstellt mit Dbdiagram.io [8]*

Die automatisch generierten Tabellen ((Siehe Abbildung 46: Datenbankdiagramm: Generierte Tabellen vom Identity Framework)) für die Authentifizierung bilden die Grundlage der Benutzerverwaltung in ASP.NET Core Identity. Zusätzlich bieten sie die Möglichkeit, erweiterte Funktionen wie Rollenmanagement oder das Zurücksetzen von Passwörtern bei Bedarf zu integrieren, ohne dass zusätzliche Tabellen erforderlich sind.

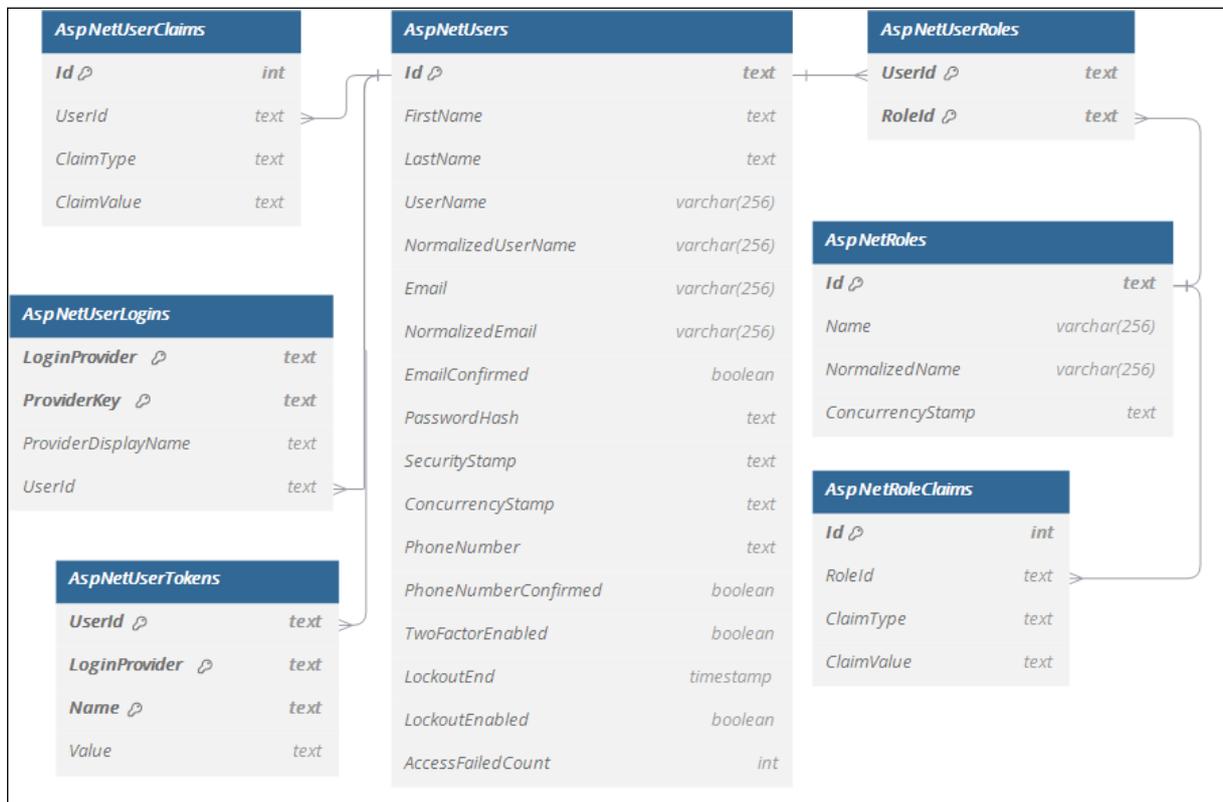


Abbildung 46: Datenbankdiagramm: Generierte Tabellen vom Identity Framework

Quelle: *Erstellt mit Dbdiagram.io [8]*

## 7.2 Infrastruktur

Die FlexSheet-Anwendung verwendet eine containerisierte Architektur, die sowohl serverseitige als auch clientseitige Komponenten umfasst. Diese Architektur ermöglicht eine hohe Flexibilität und Wartbarkeit.

### 7.2.1 Deployment

Der Deployment-Prozess (Siehe Abbildung 47: Aufbau Deployment) beginnt mit Änderungen auf dem main Branch im Repository, welcher automatisch einen Build im Buildsystem auslöst.

Der Build erstellt Container-Images für FlexSheet.Client (Frontend) und FlexSheet.Server (Backend) und lädt sie in die GitLab Container Registry hoch. Anschliessend kann man die Container manuell mit den neusten Images aus der Registry aktualisieren. Docker übernimmt die Orchestrierung und stellt sicher, dass die Container miteinander kommunizieren können. Portainer dient als Benutzeroberfläche zur Verwaltung der Container und vereinfacht die Bereitstellung, Überwachung und Verwaltung containerisierter Anwendungen.

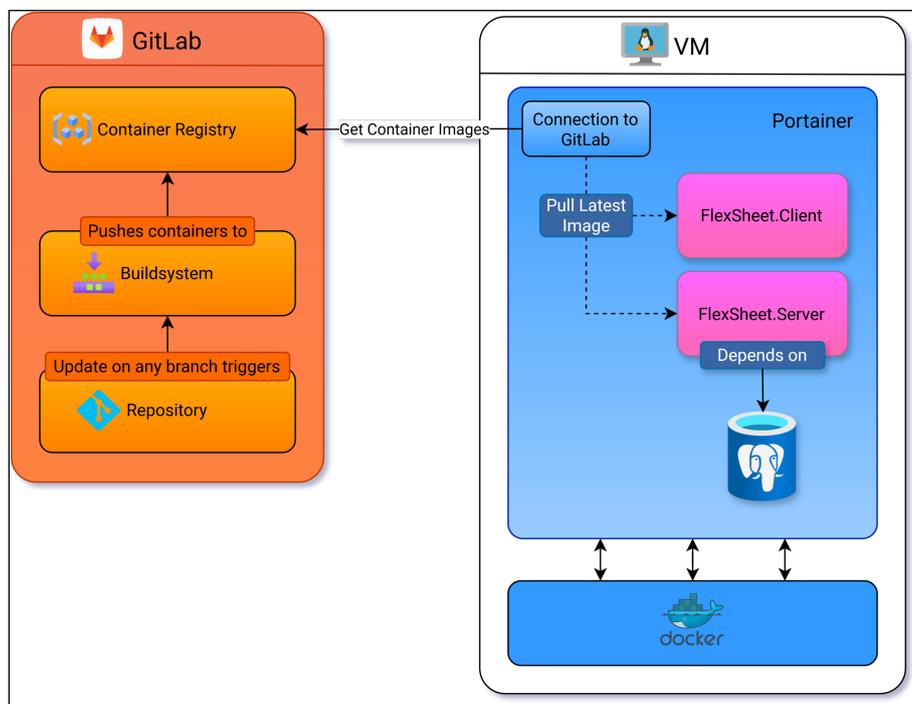


Abbildung 47: Aufbau Deployment

Quelle: Erstellt mit draw.io [10]

## 7.2.2 Kommunikationsfluss

Für die Netzwerkkommunikation (Siehe Abbildung 48: Netzwerk Kommunikation) wird ein Reverse Proxy verwendet. Der Reverse Proxy (Traefik) ist für die Weiterleitung von Anfragen zwischen dem Client, dem Server und den Benutzern verantwortlich. Traefik übernimmt die TLS-Verschlüsselung, das Zertifikatsmanagement und gewährleistet eine sichere und effiziente Weiterleitung der Anfragen an die entsprechenden Container.

1. **Initialisierung Client:** Der Benutzer greift über den Browser auf die Anwendung zu, indem er die definierte URL (`https://<URL>`) öffnet. Diese Anfrage wird von Traefik entgegengenommen und basierend auf den Weiterleitungs-Regeln an den Client-Container weitergeleitet.
2. **Auslieferung der Benutzeroberfläche:** Innerhalb des Client-Containers läuft ein Nginx-Server, der die statischen Dateien der Blazor-WebAssembly-Anwendung (JavaScript und WebAssembly-Dateien) an den Browser sendet. Der Browser lädt diese Dateien, rendert die Benutzeroberfläche und macht die Anwendung für den Benutzer verfügbar.
3. **Anfragen an das Backend:** Wenn der Benutzer in der Webanwendung Aktionen ausführt, z. B. das Abrufen von Daten oder das Speichern von Änderungen, werden API-Anfragen an die entsprechende API gesendet. Diese Anfragen werden von Traefik an den Server-Container weitergeleitet, der sie verarbeitet.
4. **Verarbeitung und Antwort:** Der Server führt die entsprechenden Datenbankoperationen durch und generiert eine Antwort. Diese Antwort wird über Traefik an den Browser zurückgesendet, wo sie von der WebAssembly-Anwendung verarbeitet und in der Benutzeroberfläche angezeigt wird.

Der zentrale Aspekt dieser Architektur ist die lose Kopplung zwischen dem Server und dem Client. Die einzige Verbindung zwischen diesen beiden Komponenten ist die URL, über die sie miteinander kommunizieren. Dieser Ansatz bringt mehrere Vorteile mit sich:

- **Flexibilität:** Durch die schwache Kopplung können der Server und der Client unabhängig voneinander entwickelt, getestet und bereitgestellt werden. Änderungen an einer Komponente (z. B. ein Server-Upgrade oder eine Neuentwicklung des Frontends) können vorgenommen werden, ohne die andere Komponente zu beeinflussen.
- **Skalierbarkeit:** Da der Client und der Server getrennt sind, können sie unabhängig voneinander skaliert werden. Beispielsweise kann bei hohem Datenverkehr der Server skaliert werden, ohne Änderungen am Client vorzunehmen, und umgekehrt.
- **Einfache Konfiguration:** Der Server benötigt lediglich die korrekte Konfiguration der CORS-Regel (Cross-Origin Resource Sharing) basierend auf der bekannten Client-URL, was die Konfiguration übersichtlich und leicht anpassbar macht. Umgekehrt muss der Client lediglich die URL des Servers kennen, um API-Anfragen an die richtige Adresse zu senden. Dieser minimalistische Konfiguration reduziert die Komplexität und gewährleistet eine klare Trennung der Verantwortlichkeiten zwischen den beiden Komponenten.

Dank dieser Architektur können Server und Client theoretisch auf unterschiedlichen Geräten oder Infrastrukturen bereitgestellt werden, ohne dass Anpassungen am Code erforderlich sind.

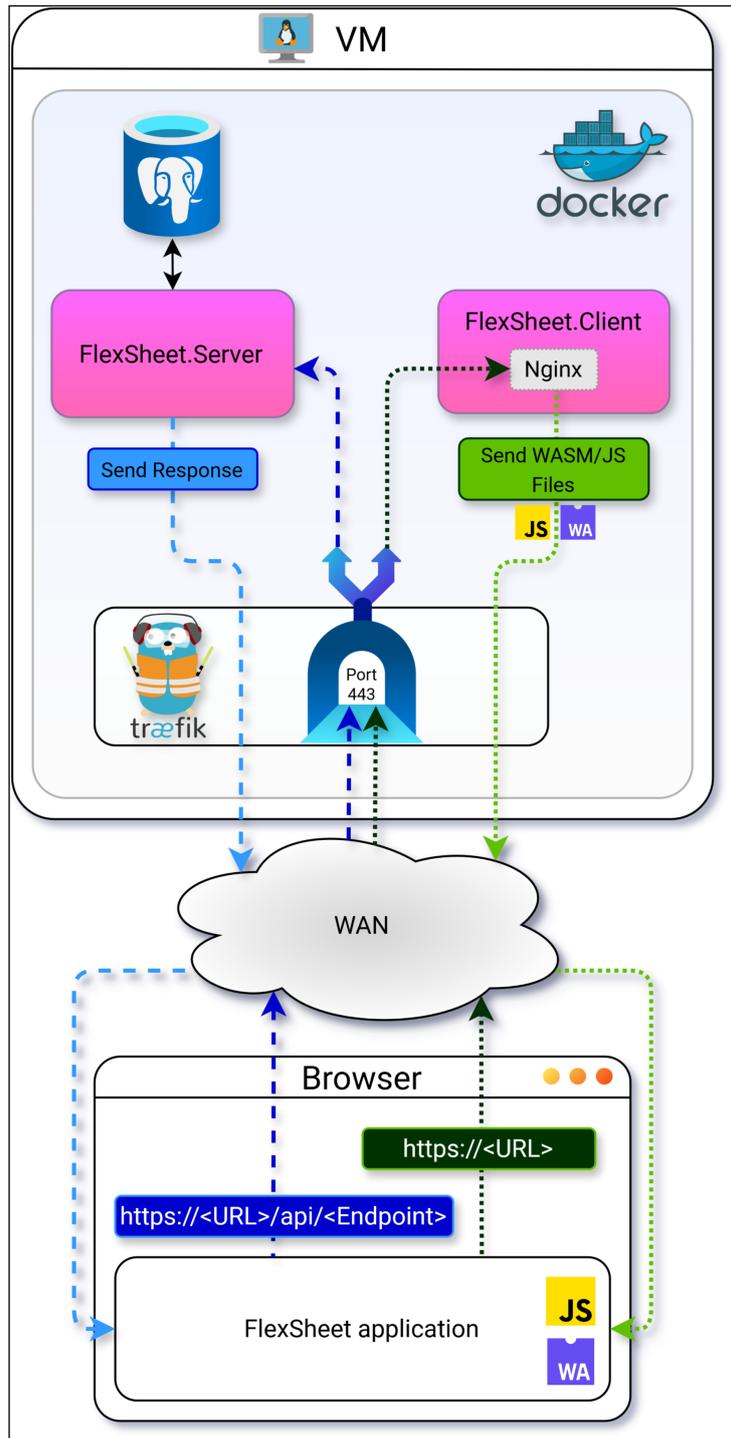


Abbildung 48: Netzwerk Kommunikation  
 Quelle: Erstellt mit draw.io [10]

## 8 Technische Umsetzung

In diesem Kapitel wird die technische Realisierung der entwickelten Webanwendung detailliert beschrieben. Dabei werden die eingesetzten Konzepte, Frameworks und Tools erläutert, die zur Implementierung der verschiedenen Funktionalitäten genutzt wurden. Zudem wird auf zentrale Aspekte wie die Generierung von Aufgaben, die Erstellung von Visualisierungen, die PDF-Generierung sowie die Authentifizierung eingegangen. Dieses Kapitel bietet somit einen umfassenden Einblick in die technische Struktur und die Herausforderungen, die während der Umsetzung gemeistert wurden.

### 8.1 Blazor Konzepte

**Dependency Injection** Dependency Injection (DI) ist ein zentraler Bestandteil des .NET-Ökosystems und wird auch in Blazor unterstützt. Dependency Injection ermöglicht es Abhängigkeiten wie Services, Konfigurationen oder Datenzugriffe effizient zu verwalten und von überall zugänglich zu machen. In Blazor wird die Dependency Injection durch die Verwendung eines Servicecontainers ermöglicht, in dem Objekte registriert und verwaltet werden können.

```
services.AddScoped<IHttpClientWithOptions, HttpClientWithOptions>();
```

Listing 4: Dependency Injection im Code

Diese Services können dann über Konstruktoren oder über den `@inject`-Tag in Komponenten (Pages) eingebunden werden (Siehe Listing 5: Einbinden von Services im Code).

```
//Property Injection
[Inject] private IRepository<ExerciseCollection> ecClient { get; set;
} = null!;

//Constructor Injection
public ExerciseCollectionDetail(IRepository<ExerciseCollection>
ecClient){ }

//Injection in Razor-Pages
@Inject IRepository<ExerciseCollection> ecClient
```

Listing 5: Einbinden von Services im Code

Die Applikation wurde vollumfänglich mit Dependency Injection aufgebaut, um keine hart kodierten Abhängigkeiten zu haben und stets modular zu bleiben.

**Localization** Die Localization (deutsch: Lokalisierung) ist ein essenzieller Bestandteil moderner Anwendungen, insbesondere wenn sie international oder für mehrere Sprachregionen zugänglich sein sollen. Blazor bietet eine integrierte Unterstützung für die Lokalisierung, welche die Lokalisierungsmechanismen von ASP.NET Core nutzt, um Ressourcen basierend auf der Kultur (Culture) des Benutzers bereitzustellen. Somit können nicht nur Übersetzungen einfach umgesetzt werden, sondern auch Variablennamen oder andere programmier spezifische Ausdrücke in eine benutzerfreundliche Sprache umgewandelt werden.

Im Rahmen dieses Projekts wurde das Konzept der Lokalisierung an mehreren Stellen genutzt, insbesondere um Variablen eines Enum-Typs für Benutzer\*innen besser verständlich zu machen, wie beispielsweise in Dropdowns oder Comboboxen. Hierfür wurde eine Resources.resx Datei erstellt (siehe Abbildung 49: Beispiel: Resources.resx Datei), die als eine Art Wörterbuch fungiert. In dieser Datei können die verschiedenen Begriffe hinterlegt und in die gewünschten

Sprachen oder Ausdrücke übersetzt werden. Dies sorgt für eine benutzerfreundliche Darstellung und erleichtert die Interaktion mit der Anwendung.

Name	Neutral Value
Addition	Addition
ArithmeticConfig	Aufgabe: Rechnen
Division	Division
ExplorerOpen	Datei Explorer öffnen
HousingTable	Tabellarisch
ImageConfig	Auflockerung: Bild

Abbildung 49: Beispiel: Resources.resx Datei

Quelle: *Aus Prototyp [28]*

Der Lokalisierungsmechanismus muss im Programm.cs konfiguriert werden und kann dann ganz einfach verwendet werden:

```
@inject IStringLocalizer<App> Localizer
<MudSelectedItem Value="type">@Localizer[type.ToString()]</MudSelectedItem>
```

Listing 6: Verwendung vom Localizer im Code

Die Lokalisierung wurde noch nicht vollständig in der Applikation eingepflegt, doch es wurde eine gute Grundlage erbaut, um die Applikation auch für andere Sprachen zugänglich machen zu können.

**Json Converter** Alle Daten, welche zwischen dem Server und dem Client kommuniziert werden, werden als eine JSON-Datenstruktur gesendet. Da die Datenstruktur polymorphe Entitäten enthält (Siehe Abbildung 7.1.4: Shared - Aufgabenkonfigurationen und 7.1.4: Shared - Aufgaben), war es notwendig, `JsonSerializerOptions` zu konfigurieren, um die korrekte Serialisierung und Deserialisierung der abstrakten Entitäten sicherzustellen. Ohne diese Konfiguration könnten die Sub-Elemente der abstrakten Entitäten nicht eindeutig zugeordnet werden.

Auf dem Model der Aufgabensammlung (Siehe Abbildung 7.1.4: Shared - Aufgabensammlung / Vorlage) befinden sich zudem zwei Eigenschaften vom Typ `NodaDateTime`, die eine spezielle Verarbeitung erfordern, um korrekt serialisiert und deserialisiert zu werden.

Um dieses Problem zu lösen, wurden zwei benutzerdefinierte `JsonConverter` implementiert. Diese überschreiben die `Read`- und `Write`-Methoden, um eine Verarbeitung der Daten zu ermöglichen. Über eine spezielle Eigenschaft im Modell wird dabei angegeben, um welchen konkreten Typ es sich handelt. Dadurch kann der Serialisierungsprozess die polymorphen Entitäten korrekt identifizieren und handhaben.

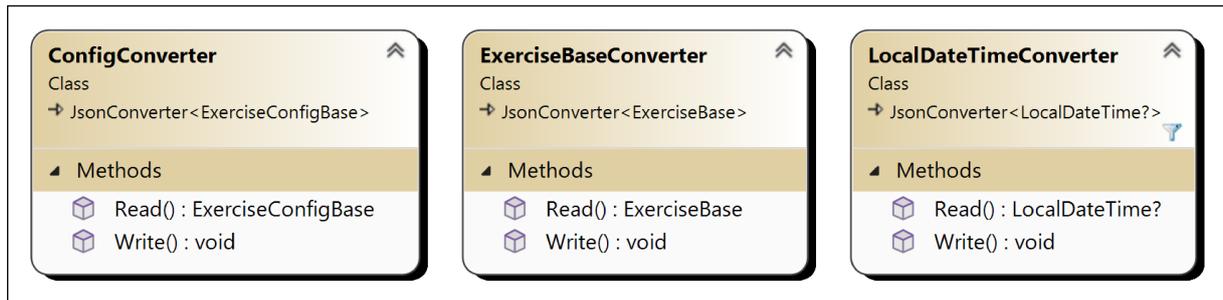


Abbildung 50: JSON Konverter

Quelle: Erstellt mit VS Class Designer [35]

Die Converter mussten für das Client-Projekt direkt dem HTTP Client übergeben werden und auf dem Server-Projekt im Program.cs registriert werden, damit auf beiden Seiten der Kommunikation erkennbar ist, wie die speziellen Typen gehandhabt werden sollen.

## 8.2 Swagger UI

Für die Dokumentation der API-Schnittstellen wurde Swagger UI eingebunden. Swagger UI ist ein Tool, welches eine visuelle und interaktive Dokumentation für RESTful APIs bereitstellt. Die Dokumentation kann automatisch aus den API-Controller und deren Attributen generiert werden und zeigt, wie sich die Schnittstellen im Backend verhalten oder welche Parameter sie erwarten. Somit konnte ohne grosse Kommunikation verständlich gemacht werden, was der Server erwartet und zurücksendet bei einer Anfrage (siehe Abbildung 51: Beispiel: Swagger UI Dokumentation). Auch ist diese Dokumentation hilfreich bei der Erweiterung dieses Projektes.

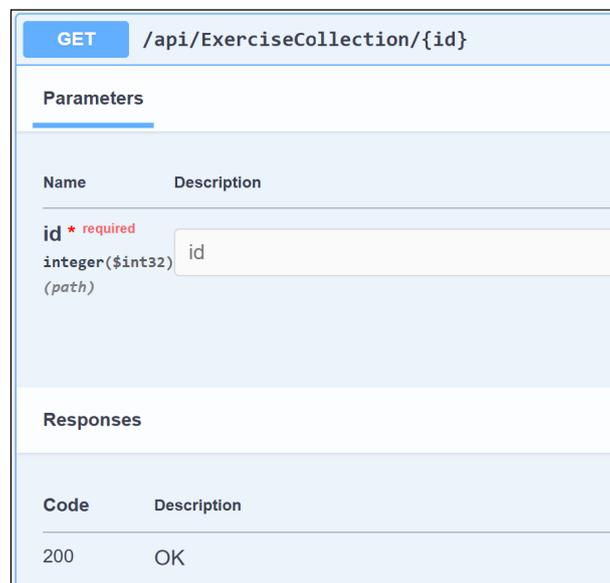


Abbildung 51: Beispiel: Swagger UI Dokumentation

Quelle: Erstellt mit Swagger UI

### 8.3 Generierung von Aufgaben

In der Applikation werden verschiedene Aufgabentypen, anhand von Benutzerdefinierten Parameter, generiert. Bis jetzt basieren alle generierbaren Aufgaben auf den folgenden elementaren Rechenoperationen:

- Addition
- Subtraktion
- Multiplikation
- Division

Dazu kommen noch folgende Visualisierungen, welche aber alle auf den oben genannten Rechenaufgaben basieren:

- Tabellen (um Additionsaufgaben zu visualisieren)
- Wendepfättchen (um Multiplikationsaufgaben zu visualisieren)
- Zahlenstrahl (um Additionsaufgaben zu visualisieren)

Für die Aufgabengenerierung existieren folgende Parameter, welche von den Benutzer\*innen definiert werden können:

- Zahlenbereich von (Lösung)
- Zahlenbereich bis (Lösung)
- Anzahl Aufgaben

Da die Aufgaben je nach gewählten Parameter, möglichst verschieden und sinnvoll generiert werden sollen, konnten nicht einfach zwei Zufallsoperanden generiert werden, da gewisse Aufgabentypen zu repetitiv oder sinnlos wären. So war es anfänglich bei der Multiplikation und Division der Fall. So wurde bei einer Multiplikation zum Beispiel zuerst der erste Operand zufällig generiert und wenn dieser grösser als 50 war, so war der zweite Operand immer 1. Auch bei der Division gab es sehr viele Aufgaben, bei welcher der Divisor immer nur 1, 2 oder die zu teilende Zahl selber war.

#### 8.3.1 Elementare Rechenoperationen

Um diese Problematik zu bekämpfen, wurden für jeden Aufgabentypen ein spezieller Algorithmus implementiert, welcher die Aufgaben möglichst divers und spannend halten soll.

**Generieren von Additionsaufgaben** Das generieren von Additionsaufgaben ist simple. Man muss nur darauf achten, dass die zwei generierten Summanden zusammengezählt nicht grösser als das Maximum des Zahlenbereichs der Lösung ist.

---

#### Algorithm 1 Generierung von Zufallszahlen für Additionsaufgaben

---

- 1: **for**  $N = 1$  to Anzahl Aufgaben **do**
  - 2:     Generiere zwei Zufallszahlen  $addend1$  und  $addend2$  im Bereich  $[2, \text{maxRange}/2]$ .
  - 3:     Rückgabe von  $(addend1, addend2)$ .
  - 4: **end for**
-

**Generieren von Subtraktionsaufgaben** Ähnlich funktioniert das Prinzip für die Subtraktionsaufgaben. Hierbei muss man nur darauf achten, dass der Minuend im Zahlenbereich liegt und der Subtrahend nicht grösser als der Minuend ist.

---

**Algorithm 2** Generierung von Subtraktionsaufgaben

---

- 1: **for**  $N = 1$  to Anzahl Aufgaben **do**
  - 2:   Generiere eine Zufallszahl *minuend* im Bereich  $[2, \text{maxRange}]$ .
  - 3:   Generiere eine Zufallszahl *subtrahend* im Bereich  $[2, \text{minuend}]$ .
  - 4:   Rückgabe von  $(\text{minuend}, \text{subtrahend})$ .
  - 5: **end for**
- 

**Generieren von Multiplikationsaufgaben** Für die Multiplikationsaufgaben muss beachtet werden, dass der erste Operand nicht grösser ist, als die Wurzel der maximalen Lösung. So kann der zweite Operand aus dem Bereich 2 bis (Maximale Lösung / Erster Operand) gewählt werden und die Lösung ist immer kleiner als die Maximale Lösung.

---

**Algorithm 3** Generierung von Multiplikationsaufgaben

---

- 1: **for**  $N = 1$  to Anzahl Aufgaben **do**
  - 2:   Generiere eine Zufallszahl *leftOperand* im Bereich  $[2, \sqrt{\text{maxRange}}]$ .
  - 3:   Generiere eine Zufallszahl *rightOperand* im Bereich  $[2, \frac{\text{maxRange}}{\text{leftOperand}}]$ .
  - 4:   Rückgabe von  $(\text{leftOperand}, \text{rightOperand})$ .
  - 5: **end for**
- 

**Generieren von Divisionsaufgaben** Bei der Generierung von Divisionsaufgaben gab es am meisten Probleme. Oftmals war der Divisor eine 2, da der generierte Dividend nur durch 1, 2 oder sich selbst teilbar war (Teilen durch 1 und sich selbst ist nicht spannend). Deshalb wurde eine Bedingung gesetzt, dass zu 90%-Wahrscheinlichkeit die Zahl 2 als Divisor vermieden wird.

Eine zweite Schwierigkeit war, dass bei gewissen Zahlen ein grosser Divisor gewählt wurde, obwohl der Dividend auch kleinere Divisoren hatte und dies vor allem in der Primarschule passendere Aufgaben generieren würde. Deshalb wurde definiert, dass mit einer 40%-Wahrscheinlichkeit, ein Divisor gewählt wird welcher nicht grösser als die Quadratwurzel des Maximal erlaubten Dividenden (Lösung) ist. Mit diesen zwei Ansätzen konnte behoben werden, dass ein unpassender Divisor gewählt wurde.

Dieser Algorithmus ist sicherlich nicht perfekt, doch generiert passende Aufgaben, ohne sich zu oft wiederholen zu müssen.

**Algorithm 4** Generierung von Divisionsaufgaben

---

```

1: for  $N = 1$  to Anzahl Aufgaben do
2:   Wähle zufällig, ob die Zahl 2 als Divisor vermieden werden soll (90% Wahrscheinlichkeit).
3:   Wähle zufällig, ob der Divisor auf Werte bis  $\sqrt{\text{maxRange}}$  eingeschränkt wird (40% Wahrscheinlichkeit).
4:   Generiere einen Divisor entsprechend den Bedingungen:
5:   if Vermeidung von 2 aktiv then
6:     Generiere einen Divisor im Bereich  $[2, \sqrt{\text{maxRange}}]$  oder  $[2, \frac{\text{maxRange}}{2}]$ .
7:     while Divisor = 2 do
8:       Generiere einen neuen Divisor.
9:     end while
10:  else
11:    Generiere einen Divisor im Bereich  $[2, \sqrt{\text{maxRange}}]$  oder  $[2, \frac{\text{maxRange}}{2}]$ .
12:  end if
13:  Generiere einen Quotienten im Bereich  $[2, \frac{\text{maxRange}}{\text{Divisor}}]$ .
14:  Berechne den Dividenden = Divisor * Quotient.
15:  if Dividenden  $\neq$  Divisor und Dividenden  $\leq$  maxRange then
16:    Rückgabe von (Dividenden, Divisor).
17:  else
18:    Wiederhole ab Schritt 1.
19:  end if
20: end for

```

---

**8.3.2 Visualisierungen**

Für die Visualisierungen werden die oben genannten Algorithmen verwendet, um die hinterlegten Rechenaufgaben zu generieren. Auf diesen Rechenaufgaben mussten jedoch je nach Typ der Visualisierung noch weitere Parameter dynamisch definiert werden.

**Generieren von Tabellen** Bei der Visualisierung als Tabelle muss darauf geachtet werden, dass die generierte Tabelle nicht zu gross aber auch nicht zu klein ist. Deshalb wird nachdem die hinterlegten Rechenaufgaben generiert werden noch die Grösse der Tabelle anhand der grössten Lösung in der Menge der Aufgaben festgelegt. Hier wird darauf geachtet, dass die Tabellengrösse immer auf ein Vielfaches von 5 aufgerundet wird.

**Algorithm 5** Berechnung der fixen Tabellengrösse

---

```

1: Bestimme maxSolution:
2:   Wähle die maximale Lösung.
3: Berechne die Basisgrösse:
4:   Nimm die Quadratwurzel von maxSolution.
5:   Runde die Basisgrösse auf das nächste Vielfache von 5 auf.
6: Rückgabe der fixen Tabellengrösse.

```

---

**Generieren von Wendepfättchen** Keine speziellen Bedingungen, da diese Visualisierung direkt von den Multiplikationsaufgaben visualisiert werden kann.

**Generieren vom Zahlenstrahl** Beim Zahlenstrahl ist der Start und das Ende bereits aus den Eingaben der Benutzer\*innen bekannt. Hier soll jedoch dynamisch der Intervall des Zahlenstrahls berechnet werden, um die "Zwischenstriche" sinnvoll anzeigen zu können.

Der Intervall wird hier ganz Trivial berechnet und macht vor allem Sinn für kleinere Zahlenbereiche. Für Zahlenbereiche welche eine Spanne von grösser als 1000 haben, ist eine Visualisierung als Zahlenstrahl sowieso fraglich. Hier könnte man noch darüber diskutieren, ob man den Intervall durch die Benutzer\*innen definieren lassen will. Für die Implementation dieses Prototypen, wurde sich erst mal dagegen entschieden.

---

**Algorithm 6** Berechnung des Intervalls basierend auf der Spanne

---

```
1: Berechne die Spanne:  $range = maxRange - minRange$ .
2: if  $range \leq 10$  then
3:   Intervall = 1.
4: else if  $range \leq 50$  then
5:   Intervall = 5.
6: else if  $range \leq 100$  then
7:   Intervall = 10.
8: else if  $range \leq 500$  then
9:   Intervall = 50.
10: else
11:   Intervall = 100.
12: end if
13: Rückgabe des Intervalls.
```

---

## 8.4 PDF Generieren

Ein Herzstück der Applikation stellt die Möglichkeit dar, die generierten Aufgaben in einem PDF-Dokument zu exportieren. Dazu wurde wie bereits im Kapitel 3: Evaluierung der Umsetzungstrategie beschrieben, iText verwendet. In der API wurde dafür ein neuer Endpunkt erstellt, welcher die generierten Aufgaben entgegennimmt, das Grundgerüst für das PDF vorbereitet und die einzelnen Aufgaben in das PDF einfügt.

Das Grundgerüst beinhaltet einen Standard HTML5 Skeleton, welcher die allgemein benötigten CSS Klassen und Attribute definiert, welche für das ganze Dokument gültig sind. Dazu gehören die Schriftart, Schriftgrösse und Schriftfarbe. Die Ränder für das PDF wurden ebenfalls definiert, damit die Aufgaben nicht ganz am Rande des PDF's kleben.

### 8.4.1 Visualisierung von Aufgaben

Für die Visualisierung der einzelnen Aufgaben sind die einzelnen Aufgaben selbst verantwortlich. Jeder Aufgabentyp hat eine eigene Komponente, welche den HTML-Code für die Darstellung beinhaltet. Diese Komponenten sind so aufgebaut, dass sie alle benötigten Informationen und Parameter entgegen nehmen. Auch werden alle spezifischen CSS-Klassen und Attribute in der Komponente definiert. Somit wird der HTML-Code für die Darstellung der Aufgabe an einem Ort definiert. Da alles benötigter Code an einem Ort gebündelt wird, wird die Wartung und Erweiterung der Aufgabentypen vereinfacht. Insbesondere wird dadurch verhindert, dass eine Änderung an einer Komponenten die Darstellung einer anderen Komponente beeinflusst.

Ein einfaches Beispiel für so eine Komponente stellen die einfachen Rechenaufgaben dar. Im oberen Teil werden die CSS-Klassen definiert, welche für die Darstellung benötigt werden gefolgt von der eigentlichen Darstellung der Aufgabe, welche die beiden Operanden und das Rechenzeichen beinhaltet. Im unteren Teil der Komponente gibt es noch einen spezifischen C#s-Code, welcher einerseits die Parameter der Komponente definiert aber auch genutzt werden kann, um die Darstellung der Aufgabe zu beeinflussen.

```

@using FlexSheet.Shared.Entities.Preset

<style>
  .arithmetic-calculation-component {
    display: flex;
    page-break-inside: avoid;
    break-inside: avoid;
    align-items: center;
    flex: 0 0 33%;
  }

  .arithmetic-calculation-component .mx-2 {
    margin: 0 8px;
  }

  .arithmetic-calculation-component .width-equation {
    width: @(Preset?.FontSizeText*4.5 ?? 70)px;
  }
</style>

<div class="arithmetic-calculation-component"
  style="margin: @((Preset?.SpaceRows ?? 1) * 8)px 0;">
  <p class="width-equation">@GetEquation()</p>
  <p class="mx-2"> = _____</p>
</div>

@code {
  [Parameter] public int FirstFactor { get; set; }
  [Parameter] public int SecondFactor { get; set; }
  [Parameter] public string Operator { get; set; }
  [Parameter] public Preset? Preset { get; set; }

  public string GetEquation() => $"{FirstFactor} {Operator} {SecondFactor}";
}

```

Listing 7: Code für Komponente für eine Rechenaufgabe

### 8.4.2 Besondere Herausforderungen

Die Entscheidung fiel bewusst auf einen PDF-Generator, der in der Lage ist, HTML-Code in PDFs umzuwandeln. Diese Wahl erlaubte es, denselben Code sowohl direkt in der App anzuzeigen als auch über iText in PDFs zu konvertieren, was ein hohes Mass an Flexibilität bot. Eine zentrale Herausforderung bestand darin, die Aufgaben möglichst dynamisch und ohne besondere Berücksichtigung der spezifischen Aufgabentypen zu generieren. Dies wurde dadurch erreicht, dass jeder Aufgabentyp eigenständig für die Darstellung der entsprechenden Aufgabe verantwortlich ist. Hierfür implementierte jeder Aufgabentyp die Methode `RenderElement(Preset? preset)`, die die Aufgabe in HTML generiert und damit sowohl für die Anzeige in der App als auch für die PDF-Erstellung nutzbar macht.

```
public abstract class ExerciseBase
{
    public abstract RenderFragment RenderElement(Preset.Preset? preset);
}
```

Listing 8: Abstrakte Methode für die PDF Generierung

Der Parameter `Preset? preset` wird speziell für die Gestaltung der PDFs verwendet. Die zugehörige Klasse `Preset` enthält alle relevanten Informationen zur PDF-Formatierung, wie beispielsweise Schriftgrösse, Schriftfarbe und Abstände. Da dieser Parameter ausschliesslich für die PDF-Generierung erforderlich ist, wurde er optional gestaltet. Dadurch können die Komponenten dieselbe Methode auch für die Darstellung in der App nutzen. Bei der Entwicklung der einzelnen Aufgaben haben wir darauf geachtet, den Code möglichst in reinem HTML zu schreiben. Die Gründe für diesen Ansatz werden im folgenden Abschnitt detailliert erläutert.

### 8.4.3 PDF Generierung Probleme

Im beruflichen Alltag zeigt sich häufig, dass die korrekte Darstellung von Elementen in PDF-Dokumenten eine Herausforderung darstellt. Das zentrale Problem liegt darin, dass verschiedene PDF-Generatoren CSS-Anweisungen nicht konsistent interpretieren und darstellen, wie es beispielsweise in Webbrowsern wie Chrome oder Firefox der Fall ist. Dadurch können Elemente, die im Browser wie gewünscht angezeigt werden, im PDF-Dokument stark abweichen. Dieser Umstand erschwert die Entwicklung erheblich, da die Fehleranalyse und -behebung komplex wird. Insbesondere ist es schwierig, die Ursachen für abweichende Darstellungen nachzuvollziehen. Hinzu kommt, dass selbst weit verbreitete CSS-Anweisungen, die in modernen Browsern als de facto Standard gelten, von PDF-Generatoren oftmals nicht oder nur fehlerhaft unterstützt werden, was zu weiteren Problemen bei der Darstellung führt.

Ein konkretes Problem trat bei der Verwendung des CSS-Attributs `flex-wrap: wrap` auf. Dieses Attribut soll bewirken, dass Flex-Elemente automatisch in die nächste Zeile umbrechen, wenn der verfügbare Platz in der aktuellen Zeile nicht ausreicht. Es konnte jedoch beobachtet werden, dass die betroffenen Elemente nicht korrekt umgebrochen wurden. Stattdessen überschritten sie sich und lagen übereinander. Dieses Problem konnte nur teilweise behoben werden, indem bestimmten Elementen feste Grössen zugewiesen wurden.

Ein weiteres Problem trat bei der Implementierung des Zahlenstrahls auf. Um eine korrekte Darstellung zu gewährleisten, mussten bestimmte Elemente des Zahlenstrahls mit festen Positionen versehen werden. Allerdings konnte iText diese fixe Positionierung nicht korrekt umsetzen, wodurch die Elemente an falschen Positionen angezeigt wurden. Zur Lösung dieses Problems wurde entschieden, den Zahlenstrahl nicht als HTML-Element in das PDF einzufügen. Stattdessen wurde ein SVG-Element generiert und in das PDF integriert. SVG-Elemente

werden von iText problemlos unterstützt und zuverlässig dargestellt.

Diese und andere Herausforderungen sowie die schwierige Fehleranalyse führten während der Entwicklung zu einem erhöhten Zeitaufwand.

## 8.5 Authentifizierung

Die FlexSheet-Anwendung implementiert die Authentifizierung basierend auf ASP.NET Core Identity[16], das sowohl serverseitige Benutzerverwaltung als auch die Bereitstellung von API-Endpunkten unterstützt. Dabei kommt eine Cookie-basiertes Authentifizierungsverfahren zum Einsatz.

### 8.5.1 Backend: Authentifizierungsinfrastruktur

**Benutzer- und API-Integration:** Die Benutzerverwaltung basiert auf der Standardklasse `IdentityUser`, die um benutzerdefinierte Felder wie `FirstName` und `LastName` erweitert wurde.[7] Diese Klasse wird durch ASP.NET Core Identity automatisch in die Datenbank integriert.

Die API-Endpunkte[33] werden über den Befehl `MapIdentityApi<ApplicationUser>()` aktiviert, um Zugriff auf Standardfunktionen wie Login, Registrierung und Passwortverwaltung zu bieten. Für die Datenbankbindung wird Entity Framework verwendet, das eine einfache Integration von Benutzerdaten in den `DataContext` ermöglicht.

#### Authentifizierungsmechanismen:

- **Cookie-basierte Authentifizierung:** Der Browser sendet bei jeder Anfrage automatisch ein Cookie, den der Server zur Identifikation des Benutzers verwendet. Vorteilhaft ist die automatische Verwaltung durch den Browser, jedoch erfordert es zusätzliche Sicherheitsmassnahmen wie den CSRF-Schutz.
- **Bearer Token:** Der Client erhält ein Token (z. B. JWT), das explizit im `Authorization-Header` mitgesendet wird. Dies ermöglicht eine plattformunabhängige und stateless Kommunikation, erfordert jedoch eine sichere Speicherung der Tokens im Client.

Microsoft empfiehlt für browserbasierte Anwendungen den Cookie-Ansatz, da dieser von Browsern automatisch verwaltet wird und eine höhere Sicherheit bietet.[23]

### 8.5.2 Frontend: Authentifizierungsstatus und Infrastruktur

**State Management:** Im Frontend überwacht die Klasse `AuthenticationStateProvider` den Authentifizierungsstatus.[2] Sie bietet folgende Kernfunktionen:

- `GetAuthenticationStateAsync`: Lädt asynchron den aktuellen Status.
- `AuthenticationStateChanged`: Event, das Statusänderungen anzeigt.
- `NotifyAuthenticationStateChanged`: Methode, um Änderungen an den Status zu propagieren.

**HTTP-Kommunikation:** Ein `HttpClient` wird verwendet, um API-Anfragen zu senden. Damit Cookies korrekt übertragen werden, wird ein `DelegatingHandler` registriert, der sicherstellt, dass Browser-spezifische Credentials (deutsch: Zugangsdaten) bei jedem Request angefügt werden.[17]

**Login, Registrierung und Logout:** Spezifische Endpunkte wie `/api/login`, `/api/register` und `/api/logout` werden im Backend bereitgestellt. Diese Methoden aktualisieren den Authentifizierungsstatus im Frontend und lösen entsprechende Änderungen in der UI aus.

### 8.5.3 Seitenabsicherung und Zugriffskontrolle

**Dynamische UI-Anpassung:** Blazor bietet mit der `AuthorizeView`-Komponente[18] eine einfache Möglichkeit Inhalte basierend auf dem Authentifizierungsstatus ein- oder auszublenden. Dies wird beispielsweise verwendet um Navigationselemente oder geschützte Seiten nur authentifizierten Nutzern zugänglich zu machen.

```
1 <AuthorizeView>
2   <Authorized>
3     <NavMenu />
4   </Authorized>
5   <NotAuthorized>
6     <p>Bitte melden Sie sich an, um auf diese Inhalte zuzugreifen.</p>
7   </NotAuthorized>
8 </AuthorizeView>
```

Listing 9: Dynamische UI-Anpassung anhand Authentifizierungsstatus

**API-Sicherheit:** Alle API-Endpunkte werden standardmässig durch `RequireAuthorization()` geschützt. Dies verhindert unbefugte Zugriffe, auch wenn clientseitige Restriktionen umgangen werden. Zusätzlich wurde ein Logout-Endpunkt manuell implementiert[18], um sicherzustellen, dass der Benutzer serverseitig korrekt abgemeldet wird und CSRF-Schutzmechanismen[21] eingehalten werden.

```
app.MapControllers().RequireAuthorization();
```

Listing 10: Absicherung der API-Endpunkte

### 8.5.4 Zusammenfassung des Authentifizierungsflusses

1. Ein Benutzer greift auf die Anwendung zu und lädt die Client-Dateien über den Browser.
2. Für das Login sendet der Client Benutzerdaten an den APIs-Endpunkt `/api/login`.
3. Der Server validiert die Anmeldedaten und gibt bei Erfolg ein Cookie zurück.
4. Bei jeder weiteren Anfrage übermittelt der Client das Cookie automatisch oder das Token im Header.
5. Geschützte API-Endpunkte verifizieren die Authentifizierung und liefern nur bei gültigen Anfragen Daten zurück.

## 9 Resultate und Ausblick

In diesem Kapitel werden die erzielten Resultate der entwickelten Applikation vorgestellt und kritisch bewertet. Dazu gehören die Umsetzung der Benutzeroberfläche, die Visualisierung von Elementen sowie die PDF-Generierung im Hinblick auf die gestellten Anforderungen. Darüber hinaus werden die Ergebnisse aus dem Usability-Test präsentiert, analysiert und mit Nutzerfeedback ergänzt. Abschliessend wird ein Ausblick gegeben, der mögliche Verbesserungen, Erweiterungen und technologische Weiterentwicklungen der Applikation aufzeigt.

### 9.1 Resultate

Dieses Kapitel präsentiert die zentralen Ergebnisse des entwickelten Prototyps. Im Fokus stehen dabei die Benutzeroberfläche der Anwendung, die Visualisierung und Bearbeitung von Aufgaben, die Generierung von PDFs sowie eine abschliessende Überprüfung, ob die funktionalen und nicht-funktionalen Anforderungen erfüllt wurden.

**Benutzeroberfläche** Die Benutzeroberfläche ist in drei Hauptbereiche gegliedert:

- Dashboard
- Aufgabensammlung
- Vorlagen

Das Dashboard (Siehe Abbildung 52: Dashboard) dient derzeit als zentraler Einstiegspunkt in die Anwendung und verweist auf die anderen Bereiche. Es verfügt aktuell über keine eigene Funktionalität.

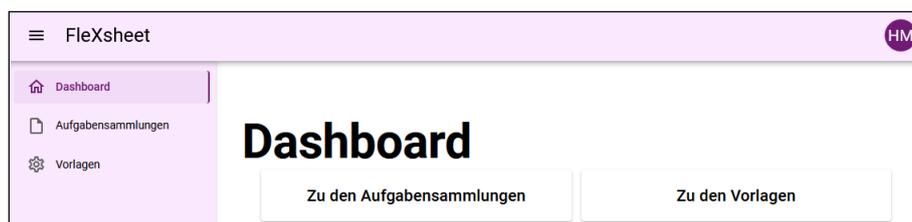


Abbildung 52: Dashboard

Quelle: Aus Prototyp [28]

Die Aufgabensammlung (Siehe Abbildung 53: Übersicht der erstellten Arbeitsblätter) bietet eine übersichtliche Darstellung aller erstellten Arbeitsblätter. Von hier aus können neue Arbeitsblätter erstellt und bestehende bearbeitet werden. Zudem ermöglicht diese Ansicht die direkte Generierung von PDFs.

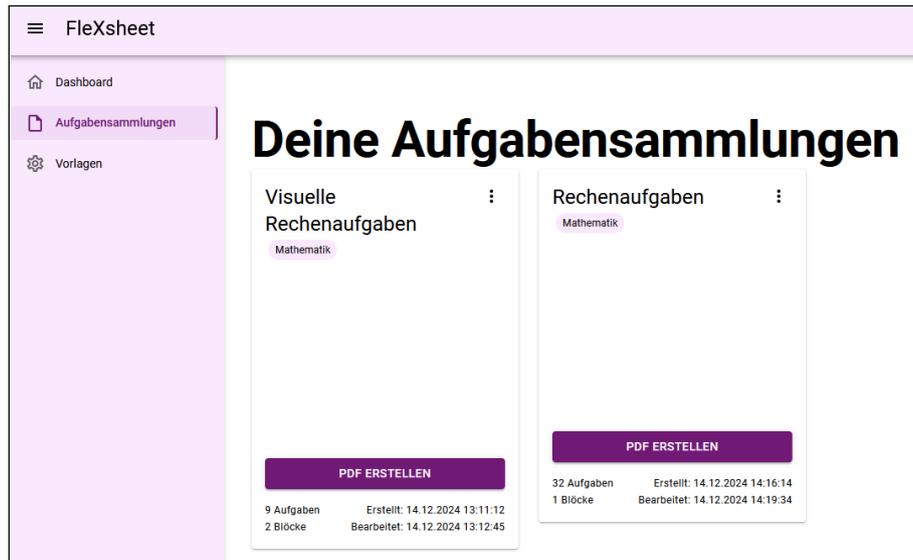


Abbildung 53: Übersicht der erstellten Arbeitsblätter

Quelle: Aus Prototyp [28]

Die Bearbeitungsansicht (Abbildung 54: Bearbeitungs- und Detailansicht eines Arbeitsblatts) entspricht der Ansicht, die beim Erstellen eines neuen Arbeitsblatts verwendet wird, mit dem Unterschied, dass bereits vorhandene Elemente angezeigt und bearbeitet werden können.

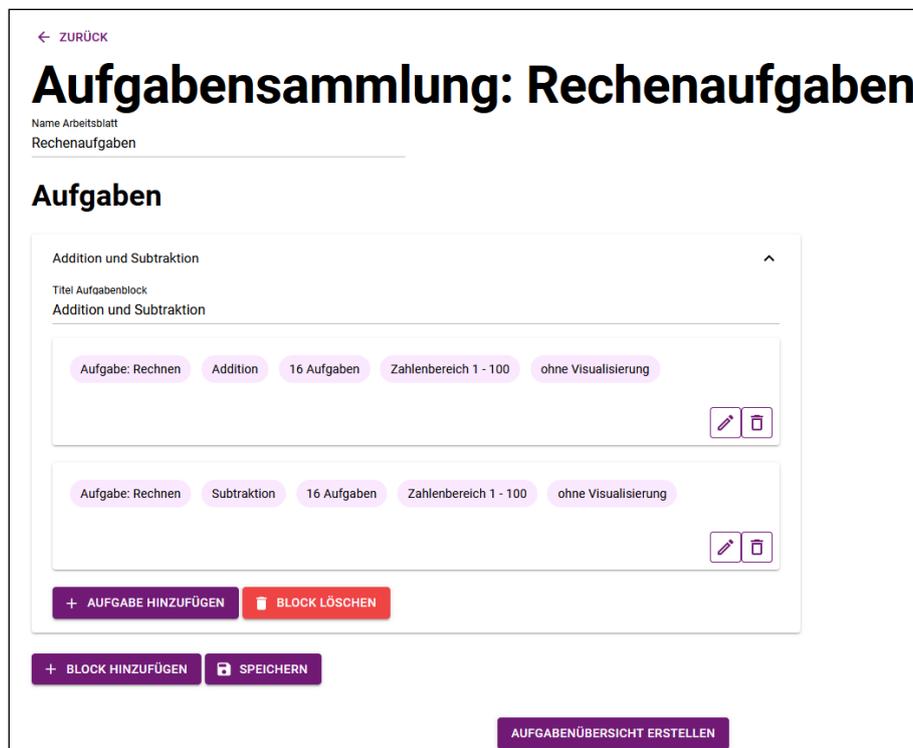


Abbildung 54: Bearbeitungs- und Detailansicht eines Arbeitsblatts

Quelle: Aus Prototyp [28]

Um eine neue Aufgabe zu erstellen, kann der Benutzer die entsprechenden Parameter in einem Dialogfeld konfigurieren (Abbildung 55: Einzelne Aufgabe). Dieser Dialog ermöglicht die

Definition der Aufgabenart mit den entsprechenden Parametern wie dem Operator, des Zahlenbereichs und Anzahl der Aufgaben.

**Aufgabe bearbeiten** ✕

Typ  
Aufgabe: Rechnen ▼

---

Operation  
Addition ▼

---

Anzahl Aufgaben  
16

---

Zahlenbereich von bis  
1 100

---

Visualisierung anzeigen

**ABBRECHEN** **HINZUFÜGEN**

Abbildung 55: Einzelne Aufgabe

Quelle: Aus Prototyp [28]

Um eine Vorschau auf die Aufgaben in der PDF-Ausgabe zu erhalten, kann eine Aufgabenübersicht (Abbildung 56: Erstellte Aufgabenübersicht) erstellt werden. Diese zeigt, wie die definierten Aufgaben auf dem PDF dargestellt werden.

← ZURÜCK

## Rechenaufgaben

**Block 1: Addition und Subtraktion**

**1. Aufgabe**

44 + 30 = ____	49 + 31 = ____	37 + 33 = ____	13 + 27 = ____
44 + 29 = ____	23 + 48 = ____	39 + 44 = ____	39 + 18 = ____
9 + 32 = ____	39 + 18 = ____	4 + 19 = ____	2 + 24 = ____
22 + 25 = ____	35 + 43 = ____	12 + 12 = ____	35 + 20 = ____

**2. Aufgabe**

19 - 10 = ____	25 - 12 = ____	46 - 41 = ____	38 - 24 = ____
33 - 17 = ____	12 - 6 = ____	90 - 31 = ____	21 - 4 = ____
30 - 7 = ____	48 - 39 = ____	65 - 8 = ____	16 - 14 = ____
34 - 20 = ____	20 - 11 = ____	37 - 17 = ____	98 - 74 = ____

Abbildung 56: Erstellte Aufgabenübersicht

Quelle: Aus Prototyp [28]

Die Verwaltung der Vorlagen erfolgt in einem separaten Bereich (Abbildung 57: Übersicht der erstellten Vorlagen). Benutzer können neue Vorlagen erstellen oder bestehende bearbeiten. Diese Vorlagen bestimmen, wie die Aufgaben auf den spezifischen Bedarf der Kinder zugeschnitten werden und dienen als Grundlage für die PDF-Generierung.

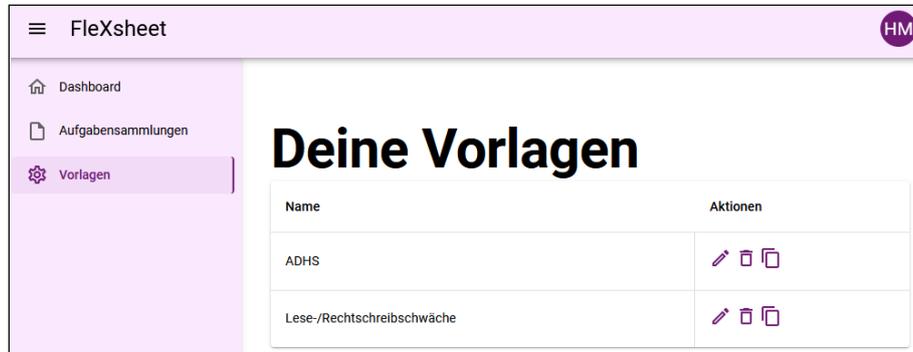


Abbildung 57: Übersicht der erstellten Vorlagen  
Quelle: Aus Prototyp [28]

Die Vorlagen können über eine eigene Maske individuell konfiguriert werden (Abbildung 58: Bearbeiten einer erstellten Vorlage).

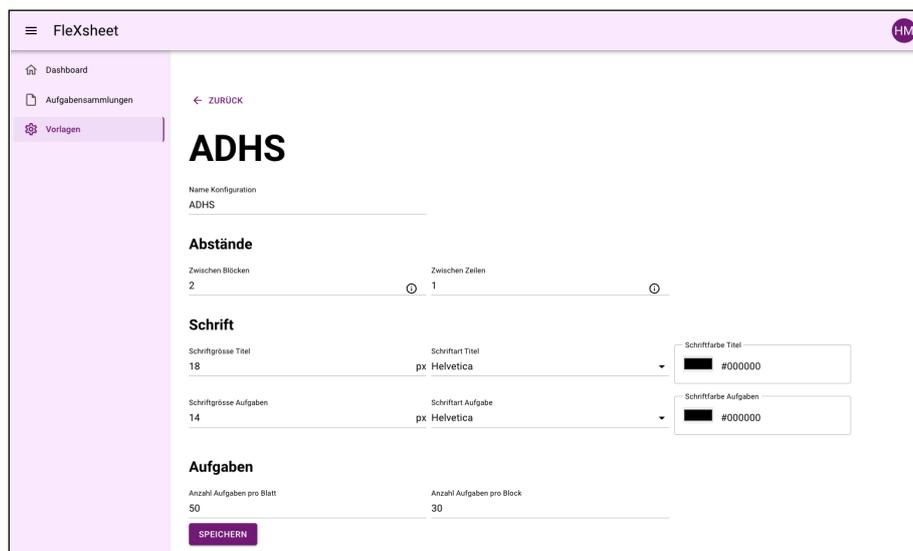


Abbildung 58: Bearbeiten einer erstellten Vorlage  
Quelle: Aus Prototyp [28]

**Visualisieren der Elemente** Im Rahmen dieser Arbeit wurden für die unterschiedlichen Rechenaufgaben Visualisierungen und Illustrationen entwickelt. Diese sollen den Schülerinnen und Schülern helfen, die Aufgaben besser zu verstehen und deren Lösungen nachzuvollziehen. Sie dienen als unterstützendes Werkzeug, um den Lösungsweg anschaulich darzustellen und die Bearbeitung zu erleichtern.

Die Visualisierungen sind in die Benutzeroberfläche integriert und können direkt bei der Erstellung einer Aufgabe ausgewählt werden. Zudem werden sie sowohl in der Benutzeroberfläche angezeigt als auch in den generierten PDF-Dokumenten übernommen. In den folgenden Abbildungen sind die verschiedenen Visualisierungsarten ersichtlich:

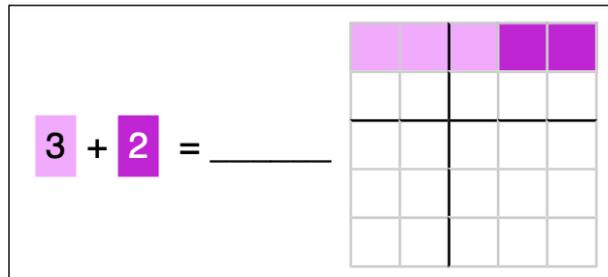


Abbildung 59: Rechenaufgabe (Addition) mit Visualisierung

Quelle: Aus Prototyp [28]

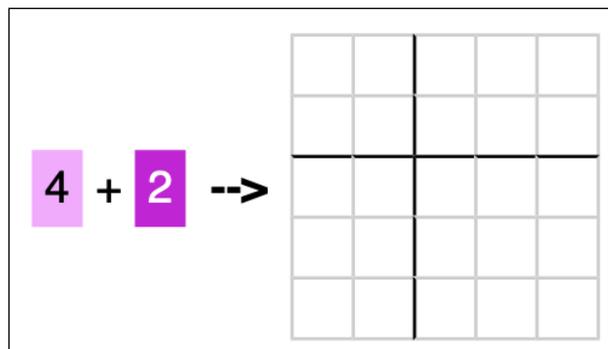


Abbildung 60: Rechenaufgabe (Addition) als Illustration

Quelle: Aus Prototyp [28]

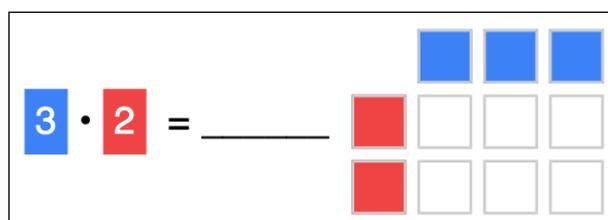


Abbildung 61: Rechenaufgabe (Multiplikation) mit Visualisierung

Quelle: Aus Prototyp [28]

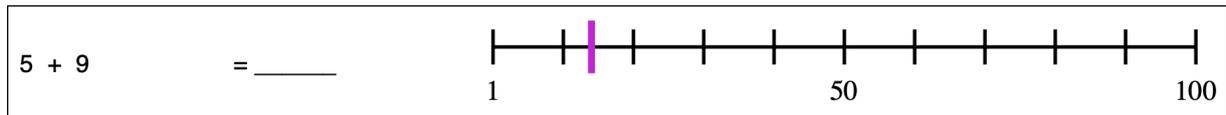


Abbildung 62: Rechenaufgabe (Addition) mit Zahlenstrahl

Quelle: *Aus Prototyp [28]*

**PDF Generierung** Der bisher zeitaufwändige und manuelle Prozess der Erstellung von Arbeitsblättern für den Unterricht konnte durch den entwickelten Prototypen erheblich vereinfacht und verschnellert werden. Benutzerinnen und Benutzer haben die Möglichkeit, die erstellten Aufgaben bequem im PDF-Format zu generieren und herunterzuladen. Diese PDF-Dokumente enthalten die erstellten Aufgaben und die entsprechenden zugehörigen Visualisierungen und Illustrationen.

Darüber hinaus bietet der Prototyp verschiedene Anpassungsmöglichkeiten welche auf das PDF angewendet werden können. Dies beinhaltet beispielsweise die Änderung der Schriftgröße, Schriftfarbe oder der Abstände zwischen Aufgaben und Elementen.

Allerdings können die Visualisierungen in den PDFs nicht immer fehlerfrei dargestellt werden. In manchen Fällen kommt es zu Überlappungen der Visualisierungen. Dies ist auf Einschränkungen des verwendeten PDF-Generators zurückzuführen, der bestimmte CSS-Eigenschaften nicht korrekt interpretiert. Eine detaillierte Beschreibung dieses Problems findet sich in Kapitel 8.4.3: PDF Generierung Probleme.

Name: \_\_\_\_\_  
 Klasse: \_\_\_\_\_

## Mathematik 1

**1. Rechne aus**

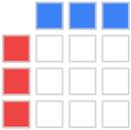
**1.1. Aufgabe**

$2 + 4 = \underline{\quad}$        $4 + 3 = \underline{\quad}$        $2 + 4 = \underline{\quad}$

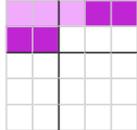
**1.2. Aufgabe**

$3 \cdot 2 = \underline{\quad}$        $2 \cdot 5 = \underline{\quad}$        $2 \cdot 5 = \underline{\quad}$

**1.3. Aufgabe**

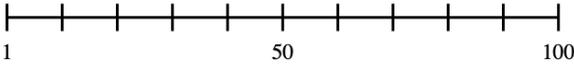
$3 \cdot 3 = \underline{\quad}$ 


**1.4. Aufgabe**

$3 + 4 = \underline{\quad}$ 


**2. Ergänze alle Darstellungen**

**2.1. Aufgabe**

$12 + 16 \rightarrow$ 


**2.2. Aufgabe**

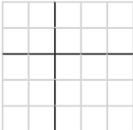
$4 + 3 \rightarrow$ 


Abbildung 63: Generierte PDF mit Rechenaufgaben und Visualisierungen  
 Quelle: Aus Prototyp [28]

**Funktionale und Nicht-Funktionale Anforderungen** Im Rahmen dieser Arbeit konnte das MVP (Siehe Kapitel 5.1.1: Primary Features) vollständig mit einzelnen, kleineren Mängeln bei den Akzeptanzkriterien erfüllt werden. Durch die komplexe Natur der Implementation und den daraus resultierenden zeitlichen Gründen, wurden die Secondary Features (Siehe Kapitel 5.1.2: Secondary Features) nicht begonnen.

Folgende User Stories enthalten in der Umsetzung Mängel:

ID	Bezeichnung	Mängel
US-3	Typen von Aufgaben 1	Die Konfiguration von den Auflockerungen (Text und Bild) wurde per Aufgabendialog möglich gemacht. Es existieren jedoch keine Formatierungsmöglichkeiten. Das Bild wird momentan noch nicht persistiert und kann somit nicht im PDF angezeigt werden. Da die Priorität auf dem Anzeigen der Rechenaufgaben und Visualisierungen lag, wurde dies vernachlässigt.
US-6	Generieren von Arbeitsblättern als PDF	Das PDF kann erfolgreich generiert und im Browser heruntergeladen werden. Die Aufgaben, ausser Bild und Text, können generiert und visualisiert werden. Es existiert jedoch nicht die Möglichkeit die Aufgaben farblich auseinanderzuhalten (Schattierung von Blöcken / Aufgaben) oder das PDF mit einer anderen Hintergrundfarbe zu erstellen. Aus zeitlichen Gründen wurde dies vernachlässigt, da auch niedrige Priorität im Prototyp.

Tabelle 19: Mängel an Primary Features

In der folgenden Tabelle 20: Evaluation der Einhaltung der Non-Functional Requirements wird die Einhaltung der definierten Non-Functional Requirements (Siehe Kapitel 5.2: Nicht-funktionale Anforderungen) dargestellt. Die Tests wurden manuell durchgeführt und die Ergebnisse entsprechend verlinkt.

ID	Beschreibung	Evaluation
NFR-1	Die Codebasis muss modular und erweiterbar gestaltet sein, um zukünftige Anpassungen und Weiterentwicklungen ohne grossen Aufwand zu ermöglichen.	Die Einhaltung der SOLID-Prinzipien wurde durch gezielte Code Reviews sichergestellt und bestätigt. Die Codebasis ist modular und erweiterbar.
NFR-2	Die Dauer zur Generierung eines druckfähigen Arbeitsblatts aus der HTML-Ansicht soll maximal 5 Sekunden betragen.	Die Generierungszeiten wurden in manuellen Tests gemessen (Siehe A.1: Durchgeführte Tests:). 90% der Tests erfüllten die Anforderung. Die Erstellung von 250 Aufgaben mit Visualisierung dauerte 11.63s und stellt einen Extremfall dar, der ausserhalb der üblichen Nutzung liegt.

ID	Beschreibung	Evaluation
<b>NFR-3</b>	Benutzer sollen unmittelbar nach der Eingabe über fehlerhafte Eingaben informiert werden, um die Integrität der Daten zu gewährleisten.	Alle Eingabefelder wurden mit Validierungsregeln versehen. Tests bestätigten, dass Fehlermeldungen klar, präzise und kontextbezogen angezeigt werden (Siehe A.1: Durchgeführte Tests:).
<b>NFR-4</b>	Die Benutzeroberfläche darf bei längeren Operationen nicht blockieren. Aktionen wie Laden und Speichern sollen asynchron im Hintergrund erfolgen.	Asynchrone Aufrufe wurden implementiert (Siehe A.1: Durchgeführte Tests:). Während die Benutzeroberfläche reaktionsfähig bleibt, fehlt beim PDF-Generierungsprozess ein visuelles Feedback (z. B. Ladeindikator).
<b>NFR-5</b>	Die Benutzeroberfläche muss so gestaltet sein, dass ein Benutzer nach maximal 20 Minuten Schulung die grundlegenden Funktionen sicher verwenden kann.	Usability-Tests ergaben, dass die Anwendung nach einer kurzen Einführung genutzt werden konnte (Siehe 9.2: Usability Test - Testergebnisse). Eine Benutzeranleitung und Optimierungen könnten die Anforderung vollständig erfüllen.
<b>NFR-6</b>	Die Applikation soll auf verschiedenen Systemen mit minimaler Konfiguration und ohne Quellcode-Änderungen lauffähig sein.	Die Portabilität wurde durch Docker-Container und Docker Compose sichergestellt. Tests auf Windows, Linux und macOS bestätigten die Übertragbarkeit (Siehe A.1: Durchgeführte Tests:).
<b>NFR-7</b>	Alle Daten zwischen Frontend und Backend müssen verschlüsselt übertragen werden, um Vertraulichkeit und Integrität sicherzustellen.	Die gesamte Kommunikation erfolgt über HTTPS. Unsichere HTTP-Anfragen wurden erfolgreich blockiert (Siehe A.1: Durchgeführte Tests:).
<b>NFR-8</b>	Der Zugriff zu den Arbeitsblättern muss ausschliesslich authentifizierten Benutzern möglich sein.	Die Authentifizierung wurde erfolgreich mit ASP.NET Core Identity implementiert. Nicht autorisierte Anfragen führen zu einem 401 Unauthorized-Statuscode (Siehe A.1: Durchgeführte Tests:).

Tabelle 20: Evaluation der Einhaltung der Non-Functional Requirements

## 9.2 Usability Test - Testergebnisse

Für die Durchführung der Usability Tests wurde der aktuelle Stand des Prototyps nach der Implementationsphase genutzt. Um eine realistische Umgebung zu gestalten, wurde die Applikation auf einem Testserver betrieben, welcher von der OST zur Verfügung gestellt wurde. Die Datenbank auf dem Testserver wurde dabei vor jeder Durchführung komplett gelöscht, um keine Hilfestellungen zu präsentieren. Über einen Laptop der Testperson wurde die Applikation über den Google Chrome Browser geladen um die Durchführung zu starten.

**Teilnehmerprofil** Ursprünglich war geplant, dass die Themenstellerin bei den Usability Tests dabei sein sollte, da ihre Reflexion und Eindrücke priorisiert werden sollte. Leider war die Themenstellerin über einen längeren Zeitraum krankheitshalber unerreichbar und konnte nicht an den Tests teilnehmen.

Die Usability Tests wurden dennoch mit zwei weiteren Personen durchgeführt.

Die Testpersonen sind zwischen 25 und 32 Jahre alt und haben beide Erfahrungen mit dem Unterrichten von Schülern. Sie absolvieren beide noch eine Pädagogische Hochschule und eine davon arbeitet nebenbei, als Unterstützung für eine Primarlehrerin. Durch ihre junge Natur haben beide Erfahrungen mit dem Umgang von Applikationen, sind jedoch nicht speziell technisch versiert und eignen sich damit perfekt als Testperson für die Usability-Tests.

**Beobachtungen** In diesem Abschnitt geht es um die Erkenntnisse, welche bei der Beobachtung der Testpersonen, während der Durchführung, aufgekommen sind.

Bezeichnung	Beschreibung
<b>Platzierung der Buttons</b>	Über die ganze Applikation hinweg, sind die Knöpfe nicht gut platziert. Vor allem die Knöpfe zum Hinzufügen verschwinden nach unten, wenn zu viele Elemente auf dem Bildschirm sind. Die Testpersonen müssen dann nach unten scrollen, was zur Verwirrung führt. Auch die Zurück-Knöpfe waren nicht direkt klar und es wurde die Zurück-Funktion des Browsers verwendet.
<b>Aufgabentypen unklar</b>	Vor allem bei Aufgabentypen, welche ein visuelles Element beinhalten, wurde nicht direkt verstanden was diese schlussendlich wirklich kreieren. Aus den Beschreibungen der Aufgabentypen wussten die Nutzer*innen nicht, ob dies der Aufgabentyp ist, den sie brauchen.
<b>Unklarheiten beim Zahlenbereich</b>	Während der Beobachtung wurde festgestellt, dass den Nutzer*innen unklar ist ob der definierbare Zahlenbereich die Lösung oder die einzelnen Werte in einer Aufgabe definieren.
<b>Workflow unklar</b>	Es war unklar, dass eine Aufgabensammlung zuerst gespeichert werden muss, bevor man auf der Aufgabenübersicht dann ein PDF daraus generieren kann.
<b>Auswählen der Vorlage</b>	Bei der Auswahl der Vorlage für das Generieren einer PDF wurde nicht verstanden, dass man im Dropdown nicht automatisch eins ausgewählt hat. Die Testpersonen haben direkt nach den "Generieren"-Knopf gesucht.
<b>Edit-Modus intuitiv</b>	Die Testpersonen konnten die Editier-Funktionen stets schnell finden und gaben auch entsprechende Rückmeldung.

Bezeichnung	Beschreibung
<b>Eingewöhnungszeit</b>	Es stellte sich heraus, dass die Personen, trotz Einführung in das Ziel der Applikation, Schwierigkeiten haben sich in der Applikation zurecht zu finden. Das legte sich jedoch, sobald die Nutzer verstanden haben, wie die Applikation aufgebaut ist.

Tabelle 21: Usability Test: Beobachtungen

**Nutzerfeedback** In diesem Abschnitt geht es um die direkte Rückmeldung der Testpersonen nach Ende der Durchführung.

Bezeichnung	Beschreibung
<b>Ziel der Applikation</b>	Den Testpersonen war anfänglich nicht klar, was die Applikation wirklich machen soll und welche Funktionalitäten vorhanden sind.
<b>Ziel der Aufgaben</b>	Den Testpersonen war unverständlich was die Visualisierungen genau bedeuten. Die elementaren Rechenoperationen wurden verstanden, jedoch war unklar wieso man einer Rechenaufgabe eine Visualisierung hinzufügen kann und was dann der Unterschied ist, wenn man direkt eine Visualisierung (Tabelle / Zahlenstrahl) als Aufgabentyp definiert.
<b>Gruppierung der Vorlagenparameter</b>	Eine Testperson meldete zurück, dass die individuelle Konfiguration einzelner Parameter bei der Vorlagenerstellung als unnötig komplex empfunden wurde. Stattdessen wurde der Wunsch geäußert, Lernschwächen einfach markieren zu können, woraufhin das System automatisch die entsprechenden Parameter setzen sollte.
<b>Darstellung der Aufgabenkonfigurationen</b>	Nachdem man mehrere Aufgabenkonfigurationen definiert hat, ist nicht direkt ersichtlich, welches Element in der Liste zu welcher Aufgabenkonfiguration gehört. Da farblich und strukturell alle gleich aussehen, kam es hier zur Verwirrung.
<b>Nicht editierbare Reihenfolge der Aufgaben</b>	Wenn bei mehreren definierten Aufgaben die Reihenfolge geändert werden soll, müssen alle Aufgaben gelöscht und neu erstellt werden. Eine Bearbeitungsmodus zur Verschiebung der Aufgaben würde Abhilfe schaffen.
<b>Übersicht der Arbeitsblätter</b>	Ein einzelnes Arbeitsblatt in der Übersicht benötigt unnötig viel Platz und es gibt keine Möglichkeit Ordner zu erstellen.

Tabelle 22: Usability Test: Nutzerfeedback

**Analyse** Die Beobachtungen und das Nutzerfeedback wurden im Anschluss analysiert und als Verbesserungsvorschläge interpretiert.

Bezeichnung	Beschreibung
<b>Benutzeranleitung</b>	Da allgemein die Funktionsweise der Applikation nicht direkt verstanden wurde und während den Tests nachgeholfen werden musste, ist klar, dass es eine Benutzeranleitung benötigt. In der Benutzeranleitung muss erklärt werden, wie man mit der Applikation umgehen muss und was schlussendlich kreierte werden kann.
<b>Aufgabentypen in einer Bibliothek</b>	Da die Aufgabentypen und deren Funktion missverstanden wurden, wäre es nützlich, wenn man eine Art Bibliothek für die verschiedenen Aufgabentypen erstellt. Hier könnte man auch gleich mit Beispielen arbeiten, welche visuell zeigen, was schlussendlich generiert wird. Da das Unterscheiden der Aufgabentypen per Dropdown sowieso keine skalierbare Lösung ist, könnte man in dieser Bibliothek auch mit Gruppier- / Filter- und Suchfunktionen arbeiten.
<b>Vordefinierte Parameter für Lernschwächen</b>	Auf der Seite der Vorlagenerstellung sollte man die Parameter nach Lernschwächen gruppieren. Es soll die Möglichkeit existieren, eine Lernschwäche auszuwählen, ohne sich spezifisch auf die darunterliegenden Parameter fokussieren zu müssen. Falls der Bedarf zur Spezifikation dieser Parameter vorhanden ist, sollte jedoch trotzdem die Möglichkeit bestehen, diese anzupassen.
<b>Platzierung der Knöpfe</b>	Die Knöpfe in der Applikation sollen so angepasst werden, dass sie stets im Blickfeld der Benutzer*innen sind.
<b>Reihenfolge der Aufgaben</b>	Die Ansicht über die Blöcke und Aufgaben sollen so angepasst werden, dass Umordnen der Aufgaben ermöglicht wird.
<b>Übersicht Arbeitsblätter</b>	Die Übersicht aller Arbeitsblätter sollte erweitert werden, um mehr Möglichkeiten zur Ordnung und Ansicht zu bieten. Dies könnte zum Beispiel die Erstellung von Ordnern, Suche und Filtermöglichkeiten beinhalten.

Tabelle 23: Analyse der Beobachtung und des Nutzerfeedbacks

**Fazit** Durch die Usability-Tests wurde verdeutlicht, wie wichtig ein menschenzentrierter Designansatz ist. Durch die realen Anwendungsbeispiele, sowie das Feedback der Teilnehmer, konnten wertvolle Erkenntnisse gewonnen werden. Da es sich um eine prototypische Implementation handelt, war zu erwarten, dass gewisse Probleme auftreten werden. Es wurde jedoch nicht damit gerechnet, dass durch die Zusammenarbeit mit den Teilnehmern so wertvolle und wichtige Verbesserungsvorschläge definiert werden können. Diese Verbesserungsvorschläge sollen verwendet werden, um bei der Weiterentwicklung des Prototypen die Benutzerfreundlichkeit zu verbessern.

### 9.3 Ausblick

In diesem Abschnitt werden die geplanten Weiterentwicklungen und Verbesserungen für die FlexSheet-Applikation beschrieben. Der Fokus liegt auf der Optimierung bestehender Funktionalitäten, der Implementierung neuer Features sowie der Weiterentwicklung der Infrastruktur.

**Bestehende Anforderungen verbessern** Ein zentrales Ziel ist die Weiterentwicklung der bereits implementierten funktionalen und nicht-funktionalen Anforderungen, die aktuell nicht vollständig erfüllt sind (Siehe Kapitel 9.1: Funktionale und Nicht-Funktionale Anforderungen). Folgende Verbesserungen sind geplant:

- **Formatierungsmöglichkeiten erweitern:** Die Darstellung und Persistenz von Text- und Bildinhalten soll verbessert werden, um flexiblere und professionellere Gestaltungsoptionen zu bieten.
- **Erweiterung der PDF-Parameter:** Weitere Einstellungen wie spezifische Schriftarten, Farben und Abstände sollen hinzugefügt werden, um besser auf die individuellen Bedürfnisse der Kinder eingehen zu können.
- **Performance-Optimierung:** Die Dauer der PDF-Generierung wird insbesondere bei umfangreicheren Arbeitsblättern weiter reduziert. Zudem sollen die Ladezeiten der Applikation beim Start verbessert werden.
- **Benutzerinteraktion verbessern:**
  - Einführung präziser Ladeindikatoren für lang andauernde Prozesse.
  - Rückmeldungen für Benutzeraktionen wie Login, Speichern oder Löschen von Elementen.
  - Warnungen, wenn bei ungespeicherten Änderungen die Benutzer\*innen versuchen die Seite zu verlassen, um Datenverlust zu vermeiden.
- **Einarbeitung der Usability-Test-Ergebnisse:** Die aus Kapitel 9.2: Usability Test - Testergebnisse gewonnenen Erkenntnisse werden umgesetzt, um die Benutzerfreundlichkeit nachhaltig zu verbessern.

**Ausbau der bestehenden Applikation** Neben der Optimierung der vorhandenen Funktionalitäten gibt es zahlreiche Erweiterungen, die die FlexSheet-Applikation bereichern könnten:

- **Implementierung der Secondary Features:** Die im Kapitel 5.1.2: Secondary Features beschriebenen zusätzlichen Features sollen vollständig umgesetzt werden.
- **Aufgabenkatalog erstellen:** Ein umfassender Katalog, der alle verfügbaren Aufgabentypen beschreibt und visualisiert, soll Benutzern einen schnellen Überblick über die Möglichkeiten der Applikation bieten. Der Katalog könnte zudem Anwendungsbeispiele und PDF-Darstellungen der Aufgaben enthalten.
- **Suchfunktion:**
  - Aufgabenkatalog: Ermöglicht das gezielte Suchen nach Aufgabentypen, Kategorien oder Tags.
  - Arbeitsblätter: Schnelles Finden erstellter Arbeitsblätter basierend auf Titel, Erstellungsdatum oder anderen Attributen.
- **Dashboard verbessern:** Das Dashboard soll ausgebaut werden, um Benutzern hilfreiche Informationen und Funktionen bereitzustellen. Geplante Verbesserungen umfassen:
  - Anzeige zuletzt bearbeiteter Arbeitsblätter.
  - Vorschläge für Vorlagen basierend auf den bisherigen Arbeiten.

- Statistiken zur Nutzung bestimmter Aufgabentypen und Vorlagen.
- Schnellzugriffe für häufig verwendete Aktionen wie das Erstellen neuer Arbeitsblätter.
- **Digitales und interaktives Lösen von Arbeitsblättern:** Eine digitale Plattform soll Schülern ermöglichen, Aufgaben direkt am Gerät zu lösen. Dabei sollen folgende Funktionen integriert werden:
  - **Audiovisuelle Unterstützung:** Vorlesefunktionen und Sprachsteuerung erleichtern den Zugang zu den Aufgaben.
  - **Haptisches Feedback:** Vibrationssignale oder ähnliche Rückmeldungen fördern die Motivation und Orientierung.
  - **Interaktive Elemente:** Dynamische Visualisierungen, wie das Färben eines Hunderterfelds, machen das Lernen interaktiver und spielerischer.

### Weitere anzustrebende Verbesserungen

- **Barrierefreiheit sicherstellen:** Die Applikation soll barrierefrei gestaltet werden, basierend auf den WCAG-Richtlinien[36]. Dazu gehören:
  - Klare Farbkontraste für Benutzer mit Sehbehinderungen.
  - Unterstützung von Screenreader für die Navigation.
  - Fokus-Indikatoren für eine barrierefreie Tastatursteuerung.
- **Lokalisierung (Internationalisierung):** Die Lokalisierung soll abgeschlossen werden, um die Applikation für unterschiedliche Regionen nutzbar zu machen. Geplante Massnahmen sind:
  - Übersetzung aller UI-Elemente, Fehlermeldungen und Hilfetexte.
  - Anpassung an regionale Formate wie Datums-, Zeit- und Zahlenformate.
- **Testkonzept erweitern:** Einführung eines strukturierten Testkonzepts, das Unit- und Integrationstests umfasst, um die Codequalität und Stabilität sicherzustellen.

**Infrastruktur: Migration zu Azure DevOps** Derzeit werden Jira und GitLab für das Projektmanagement und die Versionsverwaltung genutzt. Diese Tools erfüllen die grundlegenden Anforderungen, stossen jedoch in der Praxis auf Integrationsgrenzen. Ein Wechsel zu Azure DevOps wird angestrebt, um die Effizienz der Entwicklungsprozesse zu steigern.

### Gründe für die Migration:

- **Integration:** Azure DevOps kombiniert Versionsverwaltung, CI/CD-Pipelines, Projektmanagement und Testautomatisierung in einer zentralen Plattform.
- **Konsistenz:** Durch die Verknüpfung von Work Items mit Pull Requests wird die Nachverfolgung von Aufgaben erleichtert. Zudem ermöglicht Azure DevOps eine einfache Zeiterfassung, was die Ressourcenplanung verbessert.
- **Automatisierung:** Erweiterte Build- und Release-Pipelines sorgen für eine zuverlässige Bereitstellung und Integration von Tests.
- **Nahtlose Azure-Integration:** Die Integration mit Azure-Diensten erleichtert die Bereitstellung von Anwendungen und die Verwaltung der Infrastruktur.
- **Effizienz beim Einlernen und Benutzen:** Im bisherigen Setup mit GitLab gingen bei der Einarbeitung in die Erstellung und Verwaltung von CI/CD-Pipelines sowie beim Verständnis der Release-Prozesse erhebliche Zeit verloren.

- **Teamkompetenz:** Bereits vorhandene Erfahrung im Team mit Azure DevOps reduziert die Einarbeitungszeit.

**Fazit** Die geplanten Verbesserungen und Erweiterungen zielen darauf ab, die FlexSheet-Applikation sowohl funktional als auch technisch auf das nächste Level zu bringen. Mit einer stärkeren Ausrichtung auf Benutzerfreundlichkeit, Barrierefreiheit und Automatisierung kann das Produkt seine Zielgruppe noch besser bedienen. Die Migration zu Azure DevOps wird zudem die Entwicklungsprozesse zentralisieren und langfristig die Effizienz steigern. Damit wird ein solides Fundament geschaffen, um zukünftigen Anforderungen gerecht zu werden.

## 10 Retrospektive

In diesem Kapitel wird die Arbeit reflektiert und kritisch bewertet. Dabei wird auf die Herausforderungen während der Projektumsetzung eingegangen, die gemachten Erfahrungen im Team erläutert und die erzielten Erfolge gewürdigt. Es wird analysiert, welche Schwierigkeiten aufgetreten sind, welche Lösungen gefunden wurden und welche Erkenntnisse für zukünftige Projekte gewonnen werden konnten. Abschliessend wird das Erreichte zusammengefasst und ein Fazit gezogen, um die Ergebnisse sowie die Entwicklung des Prototyps abschliessend zu beurteilen.

### 10.1 Reflexion

Die Anforderungsanalyse und der Projektstart erwiesen sich als herausfordernd, da die Aufgabenstellung zunächst sehr offen gestaltet war und in mehrere Richtungen interpretiert werden konnte. Dies führte dazu, dass der Fokus der Arbeit zu Beginn schwer festzulegen war. Es musste viel Zeit in die Spezifikation der Anforderungen gesteckt werden, was zu gewissen Zeitverlusten führte und im Team Unklarheiten aufkommen liess.

Die intensive Untersuchung zweier technischer Implementierungsansätze sowie die iterative Entwicklung der Mockups erwies sich als sehr zeitaufwendig. Dieser parallele Arbeitsprozess war jedoch essentiell, um das Verständnis der Themenstellerin für die verschiedenen Lösungsmöglichkeiten zu vertiefen und gemeinsam eine präzise Zielsetzung zu entwickeln. Dennoch hätte während dieser Analysephase mehr darauf geachtet werden sollen, dass alle Anforderungen von der Themenstellerin direkt schriftlich festgehalten werden, um Missverständnisse zu vermeiden.

Nach der Analysephase konnte zuerst extrem Effizient gearbeitet werden. Die Infrastruktur für die Applikation konnte schnell fundamental aufgesetzt werden und auch die ersten UI Skeletons waren schon frühzeitig implementiert. Jedoch wurde schnell gemerkt, dass die technische Komplexität dieser Arbeit unterschätzt wurde und man mehr Zeit für die theoretische Auseinandersetzung der einzelnen Konzepte hätte einplanen sollen. Dies war besonders bei der Erstellung des Datenmodells der Fall. Das Datenmodell durchlief mehrere Diskussionen im Team, da man an verschiedenen Teilen auf Inkonsistenz und / oder Redundanz traf. Somit wurde das Datenmodell erst später als geplant fertig. Zwischen dem Frontend und dem Backend wurde nicht genügend transparent kommuniziert, weshalb es nach der Erstellung des Datenmodells zu Konflikten im Code kam. Hier hätte man im Team besser kommunizieren und die Schnittstellen von Anfangen an konkret festhalten müssen.

Auch die Authentifizierung und das Generieren des PDF erwiesen sich als äusserst komplexe Konzepte. Gemeinsam als Team wurde hier wesentlich mehr Zeit aufgewendet, als ursprünglich eingeplant war. Dies war jedoch ein wichtiger Einsatz, da somit das Resultat und schlussendlich die Weiterentwicklung dieses Prototypen enorm gefördert wurde. Die komplexe Implementierung ermöglichte es, die technischen Kompetenzen aller Teammitglieder optimal zu nutzen und gleichzeitig das Fachwissen in verschiedenen Bereichen zu vertiefen.

Die Arbeit im Team stellte sich insgesamt als grosse Stärke heraus. Die Aufgaben konnten gut anhand der Stärken und Schwächen jedes Teammitglieds aufgeteilt werden, womit die parallele Entwicklung gefördert wurde. Auch die Kommunikation im Team, welche anfänglich noch holprig schien, wurde im Verlaufe des Projekts immer besser und effizienter.

Ein wesentlicher Kritikpunkt ist der zu ambitionierte Umfang des Projekts. Eine Reduzierung der Anforderungen hätte ermöglicht, sich stärker auf die qualitative Entwicklung des Prototyps zu konzentrieren und flexibler auf unvorhergesehene Ausfälle durch Krankheit - sowohl der

Themenstellerin als auch im Team - während der Implementierungsphase zu reagieren. Eine striktere Planung und regelmässige Reevaluierung der Projektrisiken hätten helfen können diese Herausforderungen besser zu bewältigen.

Trotz einiger Schwierigkeiten konnten viele positive Erkenntnisse gewonnen werden. Es wurde eine solide technische Basis geschaffen, State-of-the-Art-Konzepte implementiert, sich mit modernen Infrastruktur auseinandergesetzt und wertvolle Erfahrung im Team geschaffen. Der grosse Lerneffekt und die Freude an der Zusammenarbeit waren prägende Elemente dieser Arbeit und haben dazu beigetragen, dass das gesamte Team zufrieden mit dem Ergebnis ist.

## 10.2 Fazit

Das Ziel dieser Arbeit, einen Prototypen zur Unterstützung von Lehrkräften bei der Erstellung von Unterrichtsmaterialien zu entwickeln, wurde erfolgreich umgesetzt. Der Prototyp zeigt, dass Lehrpersonen bei der Erstellung von Arbeitsblättern unterstützt werden können. Durch eben diese Unterstützung können Lehrpersonen Zeit sparen und sich auf die inhaltliche Gestaltung der Arbeitsblätter konzentrieren und auf ihren eigentlichen Beruf, nämlich das Unterrichten.

Trotz des erfolgreich umgesetzten Prototypen gab es auch einige technische Herausforderungen, wie zum Beispiel die teilweise fehlerhafte Darstellung von Visualisierungen in den PDF-Dokumenten. Auch das Datenmodell und die Authentifizierung haben mehr Zeit in Anspruch genommen als ursprünglich geplant. Dies hat zu Verzögerungen in der Entwicklung geführt, welche jedoch im weiteren Verlauf gut aufgefangen werden konnten. Auch gibt es bezüglich der PDF Generierung noch einiges an Verbesserungspotenzial. So könnte man zum Beispiel dem PDF ein ansehnliches Design verpassen, welches die Lesbarkeit und das Verständnis der Arbeitsblätter verbessert.

Der entwickelte Prototyp schneidet jedoch nur einen ganz kleinen Bereich des Themas ab. Das Themengebiet ist sehr umfangreich, sodass es in Zukunft noch viele weitere Funktionen und Features geben könnte, die die Anwendung noch nützlicher und zeiteffizienter machen würden. Nichts desto trotz wurde ein solides Fundament für die Weiterentwicklung gelegt, sodass es in Zukunft nicht nur bei einem Prototypen bleibt, sondern die Anwendung auch in den Klassenzimmern Anklang findet.

Allen Widrigkeiten zum Trotz wurde mit dem Prototypen ein wichtiger Schritt in Richtung einer modernen und digitalen Unterrichtsgestaltung gemacht.

## Literaturverzeichnis

- [1] *ADHS*. URL: <https://www.adhs.info/fuer-eltern-und-angehoerige/adhs-was-ist-das/> (besucht am 27.09.2024).
- [2] *ASP.NET Core Blazor authentication state*. URL: <https://learn.microsoft.com/en-us/aspnet/core/blazor/security/authentication-state?view=aspnetcore-8.0&pivots=webassembly> (besucht am 12.10.2024).
- [3] *ASS*. URL: <https://www.neurologen-und-psiater-im-netz.org/kinder-jugendpsychiatrie-psychosomatik-und-psychotherapie/stoerungen-erkrankungen/autismus-spektrum-stoerung-ass/#:~:text=Autismus%2DSpektrum%2DSt%C3%B6rungen%20sind%20tiefgreifende,Verst%C3%A4ndnis%20sozialer%20Situationen%20gekennzeichnet%20sind.> (besucht am 27.09.2024).
- [4] *Blazor | Build client web apps with C# | .NET*. URL: <https://dotnet.microsoft.com/en-us/apps/aspnet/web-apps/blazor> (besucht am 18.10.2024).
- [5] *C# PDF Library (All-in-One Solution) | IronPDF for .NET*. URL: <https://ironpdf.com/> (besucht am 06.11.2024).
- [6] *Canva*. URL: <https://www.canva.com/> (besucht am 27.09.2024).
- [7] *Custom user data for Identity User*. URL: <https://learn.microsoft.com/en-us/aspnet/core/security/authentication/customize-identity-model?view=aspnetcore-8.0#custom-user-data> (besucht am 10.10.2024).
- [8] *Dbdiagram.io*. URL: <https://dbdiagram.io/> (besucht am 29.10.2024).
- [9] *Docker*. URL: <https://www.docker.com/> (besucht am 10.12.2024).
- [10] *draw.io*. URL: <https://www.draw.io/> (besucht am 03.12.2024).
- [11] *Dyskalkulie*. URL: <https://www.bvl-legasthenie.de/dyskalkulie.html#:~:text=Die%20Begriffe%20Dyskalkulie%20und%20Rechenst%C3%B6rung,Schwierigkeiten%20beim%20Erlernen%20des%20Rechnens.&text=Bereits%20S%C3%A4uglinge%20k%C3%Binnen%20unterscheiden%2C%20ob,ein%20gewisses%20Mengenverst%C3%A4ndnis%20angeboren%20ist.> (besucht am 27.09.2024).
- [12] *edutopia*. URL: <https://www.edutopia.org/article/ensuring-instruction-inclusive-diverse-learners> (besucht am 27.09.2024).
- [13] Eigene Erstellung. *Manuell angepasste Arbeitsblätter zur Veranschaulichung eines optimierten Layouts*. Interne Anpassungen und Optimierungen durch die Autoren für die visuelle Darstellung in der Arbeit. 2024.
- [14] *Figma*. URL: <https://www.figma.com> (besucht am 20.12.2024).
- [15] *FluentValidation*. URL: <https://docs.fluentvalidation.net/en/latest/> (besucht am 10.12.2024).
- [16] *How to use Identity to secure a Web API backend for SPAs*. URL: <https://learn.microsoft.com/en-us/aspnet/core/security/authentication/identity-api-authorization?view=aspnetcore-8.0> (besucht am 10.10.2024).
- [17] *HttpClient Message Handlers in ASP.NET Web API*. URL: <https://learn.microsoft.com/en-us/aspnet/web-api/overview/advanced/httpclient-message-handlers> (besucht am 17.10.2024).
- [18] *HttpClient Message Handlers in ASP.NET Web API*. URL: <https://learn.microsoft.com/en-us/aspnet/core/blazor/security/?view=aspnetcore-8.0&tabs=visual-studio#authorizeview-component> (besucht am 17.10.2024).

- [19] *ISO 25010*. URL: <https://docs.arc42.org/images/1-2-iso-25010-topics-en.png> (besucht am 06. 10. 2024).
- [20] *Legasthenie*. URL: <https://www.neurologen-und-psiater-im-netz.org/kinder-jugendpsychiatrie-psychosomatik-und-psychotherapie/stoerungen-erkrankungen/lese-rechtschreibstoerung--legasthenie/#:~:text=Kinder%20mit%20einer%20Lese%2DRechtschreibst%C3%B6rung,auditiven%20und%20visuellen%20Informationsverarbeitung%20beeinflussen.> (besucht am 27. 09. 2024).
- [21] *Logout Endpoint in ASP.NET Web API*. URL: <https://learn.microsoft.com/en-us/aspnet/core/blazor/security/webassembly/standalone-with-identity/?view=aspnetcore-8.0#antiforgery-support> (besucht am 12. 10. 2024).
- [22] *Lucidchart*. URL: <https://www.lucid.app> (besucht am 22. 12. 2024).
- [23] *Microsoft recommends Cookie-based authentication*. URL: <https://learn.microsoft.com/en-us/aspnet/core/blazor/security/webassembly/standalone-with-identity/?view=aspnetcore-8.0#token-authentication> (besucht am 12. 10. 2024).
- [24] *Modern .NET library for PDF document generation*. URL: <https://www.questpdf.com/> (besucht am 06. 11. 2024).
- [25] *MudBlazor - Blazor Component Library*. URL: <https://mudblazor.com/> (besucht am 18. 10. 2024).
- [26] *naset*. URL: [https://www.naset.org/fileadmin/user\\_upload/LD\\_Report/Issue\\_\\_6\\_LD\\_Effective\\_Teach\\_Strategies.pdf](https://www.naset.org/fileadmin/user_upload/LD_Report/Issue__6_LD_Effective_Teach_Strategies.pdf) (besucht am 27. 09. 2024).
- [27] *Optical Character Recognition*. URL: [https://en.wikipedia.org/wiki/Optical\\_character\\_recognition](https://en.wikipedia.org/wiki/Optical_character_recognition) (besucht am 27. 09. 2024).
- [28] Eigenes Werk aus Prototyp. *Screenshots der Applikation zur Arbeitsblatterstellung*. Erstellt mit der prototypischen Applikation im Rahmen dieser Arbeit. 2024.
- [29] *readingrockets*. URL: <https://www.readingrockets.org/topics/dyslexia/articles/accommodating-students-dyslexia-all-classroom-settings> (besucht am 27. 09. 2024).
- [30] *SMART (Projektmanagement)*. URL: [https://de.wikipedia.org/wiki/SMART\\_\(Projektmanagement\)](https://de.wikipedia.org/wiki/SMART_(Projektmanagement)) (besucht am 06. 10. 2024).
- [31] *Structurizr*. URL: <https://structurizr.com> (besucht am 06. 10. 2024).
- [32] *The leading Java and C# PDF Library SDK*. URL: <https://itextpdf.com/> (besucht am 06. 11. 2024).
- [33] *The MapIdentityApi<TUser> endpoints*. URL: <https://learn.microsoft.com/en-us/aspnet/core/security/authentication/identity-api-authorization?view=aspnetcore-8.0#the-mapidentityapituser-endpoints> (besucht am 10. 10. 2024).
- [34] Themenstellerin. *Beispielhafte Arbeitsblätter zur Analyse*. Persönliche Kommunikation per E-Mail. Nicht veröffentlicht, übermittelt am 23. September 2024. 2024.
- [35] *VS Class Designer*. URL: <https://learn.microsoft.com/en-us/visualstudio/ide/class-designer/designing-and-viewing-classes-and-types?view=vs-2022> (besucht am 20. 10. 2024).
- [36] *Web Content Accessibility Guidelines (WCAG) 2.2*. URL: <https://www.w3.org/TR/WCAG22/> (besucht am 08. 10. 2024).
- [37] *Worksheet Crafter*. URL: <https://worksheetcrafter.com/de/> (besucht am 25. 09. 2024).
- [38] *Worksheet Crafter*. URL: <https://worksheetcrafter.com/de/2015-11-arbeitsblatter-die-kindern-mit-lese-rechtschreibschwache-lsr-bzw-legasthenie-das-leben-leichter-machen> (besucht am 27. 09. 2024).

## Glossar

- .NET** .NET ist ein Software-Framework von Microsoft zur Entwicklung und Ausführung von Anwendungen. 49, 50, 51, 56, 57, 60
- ADHS** Abkürzung für Aufmerksamkeitsdefizit-Hyperaktivitätsstörung, eine neurodevelopmentale Störung, die sich durch Unaufmerksamkeit, Hyperaktivität und Impulsivität äussert. 2
- API** API (Application Programming Interface) ist eine Schnittstelle, die es Anwendungen ermöglicht, miteinander zu kommunizieren und Daten auszutauschen. 49, 50, 51, 53, 56, 60, 69, 73, 77, 80, 81, 111
- ASP.NET Core Identity** ASP.NET Core Identity ist ein Framework von Microsoft zur Verwaltung von Benutzern, Rollen und Berechtigungen in ASP.NET Core-Anwendungen. 50, 53, 67
- ASP.NET Core** ASP.NET Core ist ein plattformübergreifendes, Open-Source-Framework von Microsoft zur Entwicklung von Webanwendungen und -diensten, das auf .NET basiert. 49, 71, 80, 90
- ASP.NET Core Web API** ASP.NET Core Web API ist ein Framework von Microsoft zur Erstellung von RESTful Web APIs. 49, 58
- Authentifizierung** Unter Authentifizierung versteht man den Prozess, bei dem die Identität eines Benutzers überprüft wird, um sicherzustellen, dass er berechtigt ist, auf ein System oder eine Anwendung zuzugreifen. 5, 42, 49, 50, 58, 67, 71, 80, 81, 90, 97, 98, 106, 111
- Azure** Azure ist eine von Microsoft bereitgestellte Cloud-Computing-Plattform, die eine Vielzahl von Cloud-Diensten und -Technologien für die Entwicklung, Bereitstellung und Verwaltung von Anwendungen bietet. 95
- Azure DevOps** Azure DevOps ist eine von Microsoft bereitgestellte Plattform, die Tools für die Planung, Entwicklung, Tests und Bereitstellung von Software in einem integrierten DevOps-Workflow bietet. 95, 96
- Blazor** Blazor ist ein von Microsoft entwickeltes Framework, das es Entwicklern ermöglicht, interaktive Webanwendungen mit C# anstelle von JavaScript zu erstellen. 49, 69, 71, 81
- Build** Build ist der Prozess, bei dem der Quellcode einer Anwendung in eine ausführbare Form kompiliert und für die Ausführung vorbereitet wird. 68, 95
- Buildsystem** Buildsystem ist ein Tool oder eine Plattform, das verwendet wird, um den Build-Prozess einer Anwendung zu automatisieren und zu verwalten. 68
- C#** C# ist eine moderne, objektorientierte Programmiersprache von Microsoft, die für die .NET-Plattform entwickelt wurde und vielseitig in der Anwendungsentwicklung eingesetzt wird. 14, 49, 50, 60, 61, 65, 78
- C4** C4 ist ein Modell zur Visualisierung der Architektur von Softwareanwendungen, das aus vier Ebenen besteht: Kontext, Container, Komponente und Code. 52
- CI/CD** CI/CD steht für Continuous Integration und Continuous Deployment/Delivery und beschreibt einen Softwareentwicklungsprozess, der automatisiertes Testen, Bauen und Bereitstellen von Anwendungen ermöglicht, um schnelle und zuverlässige Updates sicherzustellen. 95
- Code Review** Code Reviews sind systematische Überprüfungen von Quellcode, die von anderen Entwicklern durchgeführt werden, um Fehler zu finden, Best Practices zu fördern und die Qualität des Codes zu verbessern. 89

- Code-Behind** Code-Behind ist ein Design Pattern, bei dem der Code für die Logik und das Verhalten einer Benutzeroberfläche in einer separaten Datei vom Design und der Struktur der Oberfläche getrennt ist. 55
- Container-Images** Container Images sind Dateien, die alle notwendigen Abhängigkeiten und Konfigurationen enthalten, um einen Container auszuführen. 68
- CORS** CORS (Cross-Origin Resource Sharing) ist ein Sicherheitsmechanismus, der es Webanwendungen ermöglicht, Ressourcen von anderen Domains anzufordern. 69
- CRUD** CRUD ist ein Akronym, das die vier grundlegenden Operationen Create, Read, Update und Delete beschreibt. 55, 60
- CSRF** CSRF (Cross-Site Request Forgery) ist ein Angriff, bei dem ein Angreifer den Browser eines angemeldeten Benutzers dazu bringt, ungewollte Aktionen auf einer vertrauenswürdigen Webseite auszuführen, ohne dass der Benutzer es bemerkt. 80, 81
- Data Context** Data Context ist ein Konzept in Entity Framework, das die Verbindung zur Datenbank verwaltet und die Datenzugriffsschicht in einer Anwendung abstrahiert. 53
- Dependency Injection** Dependency Injection (DI) ist ein Software Design Pattern, bei dem notwendige Abhängigkeiten einer Klasse von aussen bereitgestellt (injected) werden, anstatt sie intern zu instanzieren. 71
- Deployment** Deployment ist der Prozess, bei dem eine Anwendung oder Software in einer Produktionsumgebung bereitgestellt und für Endbenutzer verfügbar gemacht wird. 68
- Docker** Docker ist eine Open-Source-Plattform, die es Entwicklern ermöglicht, Anwendungen in Containern zu erstellen, zu verwalten und zu verteilen, um die Portabilität und Skalierbarkeit von Anwendungen zu verbessern. 50, 51, 68, 90, 111
- Docker Compose** Docker Compose ist ein Tool, das es Entwicklern ermöglicht, mehrere Docker-Container zu definieren und zu verwalten, die zusammenarbeiten, um eine Anwendung zu erstellen und zu betreiben. 90
- DTO** DTO (Data Transfer Object) ist ein Design Pattern, das verwendet wird, um Daten zwischen Schichten oder Komponenten einer Anwendung zu übertragen. 58
- Entitäten** Entitäten sind Objekte oder Modelle, die in einer Datenbank gespeichert und von einer Anwendung verwendet werden. 53, 55, 57, 60
- Entity Framework Core** Entity Framework Core ist ein Object-Relational Mapping (ORM) Framework von Microsoft für die .NET-Plattform. 49, 50, 65, 66
- FluentAPI** Eine Fluent API ist ein Programmierstil, bei dem Methodenaufrufe wie in einer Kette hintereinander geschrieben werden können, sodass der Code wie ein natürlicher Satz lesbar wird. 14, 15, 21, 106
- FluentValidation** FluentValidation ist eine Validierungs-Bibliothek für .NET, die es Entwicklern ermöglicht, Validierungsregeln in einer Fluent-API zu definieren. 51, 56
- FluentValidationValidator** FluentValidationValidator ist eine Klasse in .NET, die verwendet wird, um Validierungsregeln für Datenübertragungsobjekte zu definieren und anzuwenden. 56
- Framework** Ein Framework im Kontext von Webanwendungen ist ein strukturiertes Gerüst aus vorgefertigten Komponenten, Bibliotheken und Konventionen, das Entwicklern einen standardisierten Ansatz zur effizienten Erstellung von Webanwendungen bietet. 3, 49, 50, 53, 71
- GitLab** GitLab ist eine webbasierte DevOps-Plattform, die Versionsverwaltung, CI/CD, Issue-Tracking und andere Entwicklungs- und Betriebswerkzeuge bietet. 95

**GitLab Container Registry** GitLab Container Registry ist ein Docker-Registry-Service, der es Benutzern ermöglicht, Container-Images in GitLab zu speichern und zu verwalten. 68

**Google Cloud Vision** Vortrainiertes Machine Learning Modell für maschinelles Sehen von Google. 12

**HttpClient** HttpClient ist eine Klasse in .NET, die es Entwicklern ermöglicht, HTTP-Anforderungen an Web-APIs zu senden und Daten zu empfangen. 57

**HttpInterceptorService** Ein HttpInterceptorService in .NET ist ein Dienst, der HTTP-Anfragen und -Antworten abfängt und modifiziert, bevor sie gesendet oder nachdem sie empfangen werden. 57

**HTTP Repository** HTTP Repository ist ein Design Pattern, das verwendet wird, um Daten von einer Web-API abzurufen und zu speichern. 53, 57

**Identity-API-Endpunkten** API Endpunkte für die Authentifizierung und Autorisierung von Benutzern. 50

**JavaScript** JavaScript ist eine weit verbreitete, interpretierte Programmiersprache, die hauptsächlich für dynamische, clientseitige Skripte in Webbrowsern verwendet wird, aber auch serverseitig eingesetzt werden kann. 49, 69

**JSON** JSON (JavaScript Object Notation) ist ein leichtgewichtiges Datenaustauschformat, das menschen- und maschinenlesbar ist und häufig in Webanwendungen zur Übertragung von Daten verwendet wird. 57, 72

**JWT** Ein JSON Web Token (JWT) ist ein kompakter, URL-sicherer Standard zur Übertragung von Informationen zwischen Parteien als JSON-Objekt, das digital signiert werden kann und häufig zur Authentifizierung und zum sicheren Informationsaustausch in Webanwendungen verwendet wird. 80

**LINQ** Language Integrated Query (LINQ) ist eine Microsoft-Technologie, die es ermöglicht, Datenabfragen direkt in C# oder anderen .NET-Sprachen auf eine einheitliche und lesbare Weise auszuführen. 50

**Localization** Localization ist der Prozess, bei dem eine Softwareanwendung an verschiedene Sprachen und Regionen angepasst wird, um Benutzern auf der ganzen Welt eine lokalisierte Benutzererfahrung zu bieten. 71

**LRS** LRS (Lese-Rechtschreib-Schwäche) ist eine Lernstörung, die die Fähigkeit eines Menschen beeinträchtigt, Wörter zu lesen, zu schreiben und zu verstehen. 18

**main Branch** Der main Branch ist der Hauptzweig eines Git-Repositorys, der als zentrale Quelle für den Quellcode und die Änderungen dient. 68

**MathPix** Software die auf Optical Character Recognition spezialisiert ist und es ermöglicht, mathematische Formeln und Gleichungen aus Bildern oder handschriftlichen Notizen in editierbaren digitalen Text zu konvertieren.. 12

**MFA** MFA (Multi-Factor Authentication) ist ein Sicherheitsverfahren, bei dem Benutzer sich mit mehreren Authentifizierungsmethoden anmelden müssen. 50

**Microservice** Microservice ist ein Architekturmuster, bei dem eine Anwendung in kleine, unabhängige Dienste aufgeteilt wird, die zusammenarbeiten, um die Funktionalität der Anwendung bereitzustellen. 51

**Migration** Migration ist der Prozess, bei dem Daten, Anwendungen oder Systeme von einem alten auf ein neues System übertragen werden. 95, 96

- Mockup** Mockups sind schematische Darstellungen von Benutzeroberflächen, die die Struktur, das Layout und die Interaktionen einer Anwendung visualisieren. 6, 34, 112
- MudBlazor** MudBlazor ist eine Komponenten-Bibliothek für Blazor, die Material Design-Elemente implementiert und die Entwicklung von ansprechenden Benutzeroberflächen für Blazor-Anwendungen vereinfacht. 49
- MVP** Ein Minimum Viable Product (MVP) ist eine einfache Version eines Produkts, die mit den minimal notwendigen Funktionen ausgestattet ist, um es schnell zu testen und Feedback von Nutzern zu erhalten. 39, 89
- Nginx** Nginx ist ein Open-Source-Webserver und Reverse-Proxy-Server, der für die Bereitstellung von Webanwendungen und -diensten verwendet wird. 69
- OAuth2** OAuth2 ist ein Autorisierungsprotokoll, das es Benutzern ermöglicht, Drittanbieteranwendungen den Zugriff auf ihre geschützten Ressourcen zu gewähren. 50
- OCR** Technologie zur automatisierten Texterkennung in Bildern. 11, 12, 13
- Open-Source** Open-Source bezeichnet Software, deren Quellcode öffentlich zugänglich ist und von der Gemeinschaft frei genutzt, geändert und weiterentwickelt werden kann. 50
- OpenID-Connect** OpenID-Connect ist ein Authentifizierungsprotokoll, das auf OAuth2 basiert und es Benutzern ermöglicht, sich bei Webanwendungen zu authentifizieren. 50
- ORM** Object-Relational Mapping (ORM) ist eine Technik, die Objekte in einer Programmiersprache automatisch mit Datenbanktabellen verknüpft, sodass Datenbankoperationen wie Abfragen und Änderungen direkt über diese Objekte durchgeführt werden können. 49, 65
- Pipeline** Pipeline ist eine Reihe von automatisierten Schritten, die in einem CI/CD-Workflow ausgeführt werden, um Quellcode zu testen, zu bauen und bereitzustellen. 95
- Polymorphie** Polymorphie ist ein Konzept der objektorientierten Programmierung, bei dem Objekte unterschiedliche Formen annehmen können. 57
- Portainer** Portainer ist eine Open-Source-Management-Plattform für Docker-Container, die es Benutzern ermöglicht, Container zu erstellen, zu verwalten und zu überwachen. 68
- PostgreSQL** PostgreSQL ist ein relationales Datenbankmanagementsystem, das für die Speicherung und Verwaltung von Daten verwendet wird. 51
- Postman** Postman ist ein Tool, das Entwicklern ermöglicht, APIs zu testen indem es eine einfache Benutzeroberfläche für die HTTP Abfragen bietet. 111
- Proof of Concept** Ein Proof of Concept (PoC) ist ein kleines Projekt oder Experiment, das beweisen soll, dass eine bestimmte Idee oder ein Konzept technisch machbar ist und funktioniert, bevor man Zeit und Geld in die vollständige Entwicklung investiert. 21
- Pull Request** Pull Request ist eine Funktion in Versionskontrollsystemen wie Git, die es Entwicklern ermöglicht, Änderungen an einem Repository vorzuschlagen und zu überprüfen, bevor sie in den Hauptzweig (Master) übernommen werden. 95
- Razor-Komponenten** Razor-Komponenten sind wiederverwendbare UI-Bausteine in ASP.NET Core, die HTML und C#-Code kombinieren, um dynamische Webseiten-Elemente zu erstellen. 53, 55
- Release** Release ist ein Zeitpunkt, zu dem eine neue Version einer Software oder Anwendung veröffentlicht und für Benutzer verfügbar gemacht wird. 95
- Repository Pattern** Repository Pattern ist ein Design Patter, das verwendet wird, um Datenzugriffsschichten in einer Anwendung zu abstrahieren und zu kapseln. 59, 60

- RESTful** RESTful ist ein Architekturstil für die Entwicklung von Web-APIs, der auf dem REST-Prinzip basiert und es ermöglicht, Ressourcen über standardisierte HTTP-Methoden zu verwalten. 60
- Reverse Proxy** Reverse Proxy ist ein Server, der als Vermittler zwischen Client und Server fungiert und Anfragen an den richtigen Server weiterleitet. 69
- Screenreader** Ein Screenreader ist ein Hilfsmittel-Programm, das digitale Inhalte (wie Text, Bilder und Benutzeroberflächen) für sehbehinderte oder blinde Menschen vorliest oder in Brailleschrift übersetzt. 12, 95
- SHP** SHP (Schulische Heilpädagogin) ist eine spezialisierte Lehrperson, die Schülerinnen und Schüler mit besonderen Bedürfnissen unterstützt und fördert. 39, 40, 41, 42, 43
- Skeleton** Skeletons sind einfache, ungestylte Versionen von Benutzeroberflächen, die als Platzhalter für Inhalte und Layouts dienen, bis der endgültige Inhalt und das Design hinzugefügt werden.. 97
- SOLID** Prinzipien objektorientierten Designs. 89
- SQL Express** SQL Express ist eine kostenlose, leichte Version von Microsoft SQL Server, die für die Entwicklung und Bereitstellung von Anwendungen verwendet wird. 51
- SVG** SVG (Scalable Vector Graphics) ist ein Grafikformat für skalierbare Vektorgrafiken, die verlustfrei vergrößert werden können und mit Code erstellt und manipuliert werden können. 79
- Swagger UI** Swagger UI ist ein Open-Source-Werkzeug, das es Entwicklern ermöglicht, APIs zu dokumentieren, zu testen und zu visualisieren, indem es eine interaktive Benutzeroberfläche für die API-Endpunkte generiert. 5, 73
- TLS** TLS (Transport Layer Security) ist ein Verschlüsselungsprotokoll, das verwendet wird, um die Sicherheit und Integrität von Datenübertragungen im Internet zu gewährleisten. 69
- Toolbelt.Blazor** Toolbelt.Blazor ist eine Bibliothek für Blazor, die nützliche Komponenten und Dienste für die Entwicklung von Blazor-Anwendungen bereitstellt. 57
- Traefik** Traefik ist ein Open-Source-Reverse-Proxy und Load-Balancer, der für die Bereitstellung von Webanwendungen und -diensten in Containern verwendet wird. 69
- UI** UI (User Interface) ist die Benutzeroberfläche einer Softwareanwendung, die es Benutzern ermöglicht, mit der Anwendung zu interagieren und Aufgaben auszuführen. 81, 97
- Unit of Work** Unit of Work ist ein Design Pattern, das verwendet wird, um Transaktionen in einer Anwendung zu verwalten und die Konsistenz der Datenbank zu gewährleisten. 53, 59, 60
- WCAG** Die Web Content Accessibility Guidelines (WCAG) 2.2 sind internationale Richtlinien, die dabei helfen, Webinhalte zugänglicher für Menschen mit Behinderungen zu gestalten. Die Richtlinien bestehen aus drei Konformitätsstufen: Level A (Grundanforderungen), Level AA (mittleres Mass an Barrierefreiheit) und Level AAA (höchstes Mass). 95
- WebAssembly** WebAssembly ist ein Binärformat mit geringer Grösse und nahezu nativer Ausführungsgeschwindigkeit, das es ermöglicht, Code in verschiedenen Programmiersprachen zu kompilieren und im Webbrowser auszuführen. 49, 69

## Abbildungsverzeichnis

1	Überfülltes Arbeitsblatt . . . . .	2
2	Manuell überarbeitetes Arbeitsblatt . . . . .	2
3	Generiertes PDF . . . . .	4
4	Ablenkendes Bild . . . . .	7
5	Abgeklebtes Bild . . . . .	7
6	Überfülltes Aufgabenblatt . . . . .	8
7	Überarbeitetes Layout des Aufgabenblattes . . . . .	8
8	Hilfestellungen manuell hinzufügen . . . . .	9
9	Preismodelle von IronPDF . . . . .	13
10	Beispiel einer FluentAPI in QuestPDF . . . . .	15
11	Illustrationsbeispiel von Worksheet Crafter . . . . .	16
12	Illustrationsbeispiel von Worksheet Crafter . . . . .	16
13	Illustrationsbeispiel von Worksheet Crafter . . . . .	16
14	Vorlagenbeispiel von Canva . . . . .	17
15	Illustrationsbeispiel von Canva (Nicht dynamisch veränderbar) . . . . .	17
16	Wireframes Fundament: Dashboard . . . . .	23
17	Wireframes Fundament: Vorlagen . . . . .	24
18	Wireframes Fundament: Vorlagen konfigurieren . . . . .	24
19	Wireframes Variante 1: Layout . . . . .	25
20	Wireframes Variante 1: Blöcke verwalten . . . . .	26
21	Wireframes Variante 1: Aufgaben hinzufügen . . . . .	27
22	Wireframes Variante 2: Layout . . . . .	29
23	Wireframes Variante 2: Aufgaben definieren . . . . .	30
24	Wireframes Variante 2: Aufgaben werden hinzugefügt . . . . .	31
25	Sonderpädagogische Bedürfnisse . . . . .	32
26	Programmfluss Variante 2 . . . . .	33
27	Programmfluss Überarbeitet . . . . .	33
28	Erstellen eines Arbeitsblatt nach Überarbeitung . . . . .	34
29	Multiplikationsaufgabe als Wendepfättchen visualisiert . . . . .	35
30	Additionsaufgabe als Tabelle visualisiert . . . . .	35
31	ISO 25010 Qualitätscharakteristiken . . . . .	44
32	C4 Context Diagramm . . . . .	52
33	C4 Container Diagramm . . . . .	53
34	C4 Component Diagramm . . . . .	54
35	Klassendiagramm: Business Logik der Pages . . . . .	55
36	Klassendiagramm: Validatoren . . . . .	56
37	Klassendiagramm: HTTP Repository . . . . .	57
38	Klassendiagramm: Clientseitige Authentifizierung . . . . .	58
39	Klassendiagramm: Backend . . . . .	59
40	Klassendiagramm: Models (Aufgabensammlung / Vorlage) . . . . .	61
41	Klassendiagramm: Models (Aufgabenkonfigurationen) . . . . .	62
42	Klassendiagramm: Models (Aufgaben) . . . . .	63
43	Klassendiagramm: Aufgabengenerator . . . . .	64
44	Klassendiagramm: Datenkontext . . . . .	65
45	Datenbankdiagramm: Tabellen aus EF Core generiert . . . . .	66
46	Datenbankdiagramm: Generierte Tabellen vom Identity Framework . . . . .	67
47	Aufbau Deployment . . . . .	68
48	Netzwerk Kommunikation . . . . .	70

49	Beispiel: Resources.resx Datei . . . . .	72
50	JSON Konverter . . . . .	73
51	Beispiel: Swagger UI Dokumentation . . . . .	73
52	Dashboard . . . . .	82
53	Übersicht der erstellten Arbeitsblätter . . . . .	83
54	Bearbeitungs- und Detailansicht eines Arbeitsblatts . . . . .	83
55	Einzelne Aufgabe . . . . .	84
56	Erstellte Aufgabenübersicht . . . . .	84
57	Übersicht der erstellten Vorlagen . . . . .	85
58	Bearbeiten einer erstellten Vorlage . . . . .	85
59	Rechenaufgabe (Addition) mit Visualisierung . . . . .	86
60	Rechenaufgabe (Addition) als Illustration . . . . .	86
61	Rechenaufgabe (Multiplikation) mit Visualisierung . . . . .	86
62	Rechenaufgabe (Addition) mit Zahlenstrahl . . . . .	87
63	Generierte PDF mit Rechenaufgaben und Visualisierungen . . . . .	88
64	Dashboard mit Aufgabeblättern . . . . .	112
65	Erstellen eines neuen Aufgabenblattes . . . . .	112
66	Auswählen von Aufgabentypen (Rechenaufgabe) . . . . .	113
67	Eingabe von gewünschten Parametern für Rechenaufgabe . . . . .	113
68	Anzeige neu erstellte Rechenaufgabe . . . . .	114
69	Auswählen von Aufgabentypen (Illustration) . . . . .	114
70	Eingabe von gewünschten Parametern für Illustration . . . . .	115
71	Anzeige neu erstellte Illustration . . . . .	115
72	Erstellen von Rechenaufgabe mit Visualisierung . . . . .	116
73	Anzeige neu erstellte Aufgabe mit Visualisierung . . . . .	116
74	Generierte Aufgaben in Übersicht anzeigen . . . . .	117
75	Anwenden von Konfigurationen auf PDF . . . . .	118
76	Liste mit allen möglichen PDF Konfigurationen . . . . .	118
77	Bearbeiten einer PDF Konfiguration . . . . .	119

## Tabellenverzeichnis

1	Functional Requirements (US-1)	39
2	Functional Requirements (US-2)	40
3	Functional Requirements (US-3)	40
4	Functional Requirements (US-4)	41
5	Functional Requirements (US-5)	41
6	Functional Requirements (US-6)	41
7	Functional Requirements (US-7)	42
8	Functional Requirements (US-8)	42
9	Functional Requirements (US-9)	43
10	Functional Requirements (US-10)	43
11	Non Functional Requirements (NFR-1)	45
12	Non Functional Requirements (NFR-2)	45
13	Non Functional Requirements (NFR-3)	46
14	Non Functional Requirements (NFR-4)	46
15	Non Functional Requirements (NFR-5)	47
16	Non Functional Requirements (NFR-6)	47
17	Non Functional Requirements (NFR-7)	48
18	Non Functional Requirements (NFR-8)	48
19	Mängel an Primary Features	89
20	Evaluation der Einhaltung der Non-Functional Requirements	90
21	Usability Test: Beobachtungen	92
22	Usability Test: Nutzerfeedback	92
23	Analyse der Beobachtung und des Nutzerfeedbacks	93

## Algorithmen Verzeichnis

1	Generierung von Zufallszahlen für Additionsaufgaben . . . . .	74
2	Generierung von Subtraktionsaufgaben . . . . .	75
3	Generierung von Multiplikationsaufgaben . . . . .	75
4	Generierung von Divisionsaufgaben . . . . .	76
5	Berechnung der fixen Tabellengrösse . . . . .	76
6	Berechnung des Intervalls basierend auf der Spanne . . . . .	77

## A Testprotokoll

### A.1 Manuelle Tests

Die manuellen Tests werden durchgeführt, um sicherzustellen, dass die definierten nicht funktionalen Anforderungen eingehalten werden.

**Testumgebung:** Die manuellen Tests wurden in der produktiven Umgebung durchgeführt, da diese am Ende des Projektes bereinigt und frisch aufgesetzt wird. Somit ist garantiert, dass es keine Unterschiede gibt zu der Applikation, welche die Benutzer später verwenden werden.

**Durchgeführte Tests:** Zur Validierung der definierten Non-Functional Requirements (NFRs) wurden umfassende manuelle Tests durchgeführt. Die nachfolgenden Abschnitte beschreiben die Testmethodik und die erzielten Ergebnisse für jede relevante Anforderung.

- **NFR-2 (Performance):** Die Dauer der Generierung von PDFs wurde mit Arbeitsblättern unterschiedlicher Komplexität gemessen. Pro Arbeitsblatt erfolgten jeweils drei Testläufe unter Verwendung derselben Vorlage. Die Ergebnisse sind im Folgenden aufgeführt:

1. **25 Aufgaben ohne Visualisierung:** 0.53 s
2. **25 Aufgaben mit Visualisierung:** 2.45 s
3. **50 Aufgaben ohne Visualisierung:** 0.58 s
4. **50 Aufgaben mit Visualisierung:** 2.48 s
5. **75 Aufgaben ohne Visualisierung:** 0.58 s
6. **75 Aufgaben mit Visualisierung:** 2.73 s
7. **100 Aufgaben ohne Visualisierung:** 0.64 s
8. **100 Aufgaben mit Visualisierung:** 4.59 s
9. **250 Aufgaben ohne Visualisierung:** 1.06 s
10. **250 Aufgaben mit Visualisierung:** 11.63 s

Es zeigte sich, dass die Generierungsdauer primär von der Anzahl der Aufgaben und der Aktivierung von Visualisierungen beeinflusst wird. Ohne Visualisierungen bleibt die Generierung auch bei umfangreichen Arbeitsblättern effizient.

- **NFR-3 (Input-Validierung):** Die Eingabevalidierung wurde manuell getestet, um sicherzustellen, dass ungültige Eingaben erkannt und durch präzise Fehlermeldungen kommuniziert werden. Geprüft wurden die folgenden Formulare und Eingabefelder:
  - **Login und Registrierung**
  - **Arbeitsblatt-Erstellung und -Bearbeitung:**
    - \* Name des Arbeitsblatts
    - \* Titel des Aufgabenblocks
    - \* Eingabefelder für alle Aufgabentypen: Rechnen, Tabelle, Zahlenstrahl, Text und Bild
  - **Vorlagen-Erstellung und -Bearbeitung:**
    - \* Name der Vorlage
    - \* Abstandsparameter
    - \* Schriftart und -grösse
    - \* Aufgabeninhalte
  - **PDF-Generierung:**

- \* Auswahl der Vorlage
- \* Eingabefelder zur Individualisierung der Vorlage

Die Tests zeigten, dass fehlerhafte Eingaben zuverlässig abgefangen und dem Benutzer klar verständliche Fehlermeldungen angezeigt werden.

- **NFR-4 (Asynchrone Handlungen):** Zur Gewährleistung einer reaktionsfähigen Benutzeroberfläche während laufender Hintergrundprozesse wurden folgende Operationen überprüft:
  - Authentifizierung (Login und Logout)
  - Laden der Übersicht aller Arbeitsblätter
  - Erstellung der PDFs

Alle asynchronen Prozesse liefen im Hintergrund ab, ohne die Benutzeroberfläche zu blockieren. Allerdings fehlte in einigen Fällen visuelles Feedback, wie etwa ein Ladeindikator welcher anzeigt, dass im Hintergrund ein Prozess noch im Gange ist.

- **NFR-6 (Plattformunabhängigkeit):** Die Applikation wurde auf den Betriebssystemen Windows, Linux und macOS getestet. Die Docker-Container liessen sich mit minimaler Konfiguration starten, und die Übertragbarkeit der persistierten Datenbank wurde erfolgreich verifiziert. Es war lediglich erforderlich, den Datenbankordner zu verschieben.
- **NFR-7 (Sichere Datenübertragung):** Die Sicherheit der Datenübertragung wurde mit dem Tool Postman getestet. Alle API-Aufrufe erfolgten ausschliesslich über HTTP, unsichere Verbindungen wurden korrekt blockiert. Zusätzlich wurde sichergestellt, dass die gesamte Applikation nur über HTTP aufgerufen werden kann.
- **NFR-8 (Authentifizierung):** Die Authentifizierung wurde durch gezielte Tests überprüft. API-Endpunkte wurden manuell mit nicht authentifizierten Benutzern getestet, wobei stets der Statuscode 401 (Unauthorized) zurückgegeben wurde. Im Frontend wurde sichergestellt, dass nicht authentifizierte Benutzer automatisch auf die Login-Seite umgeleitet werden.

## B Mockups

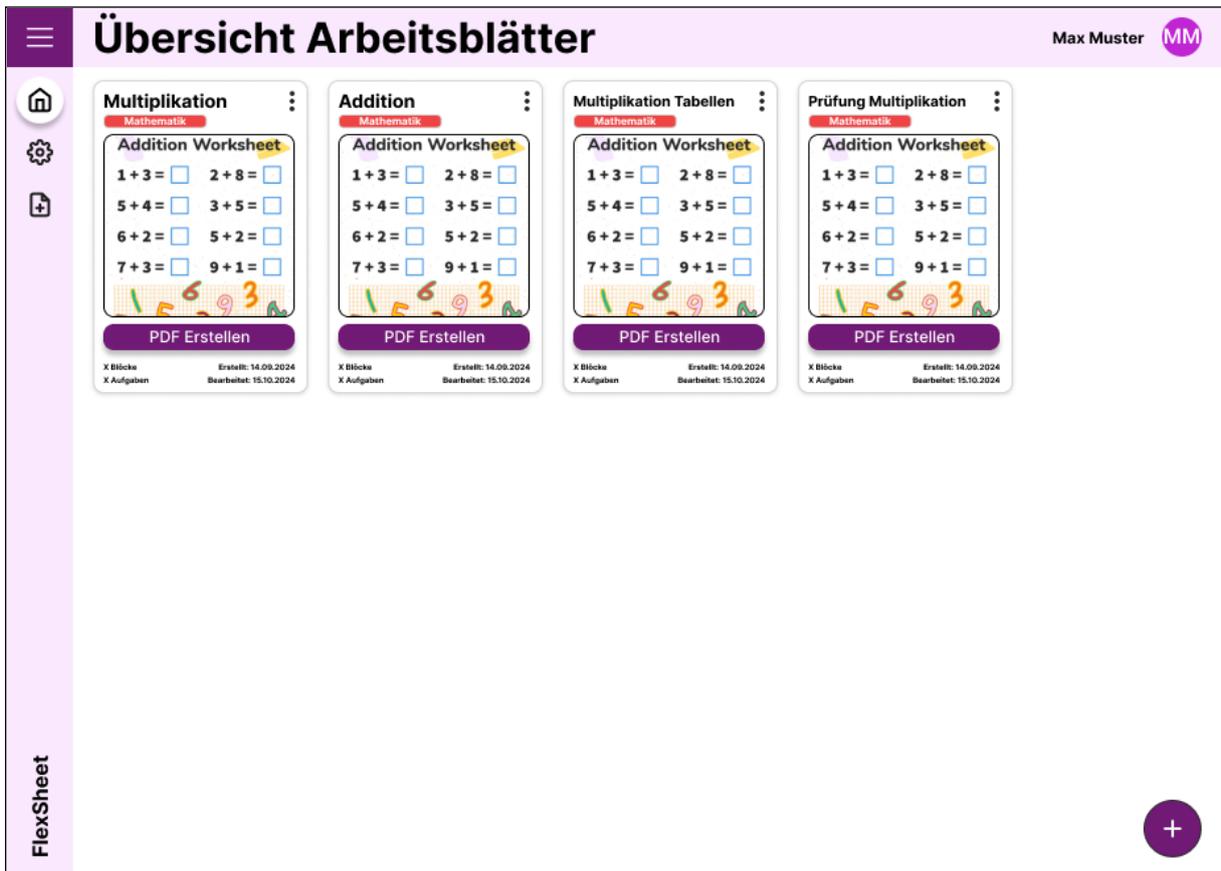


Abbildung 64: Dashboard mit Aufgabeböckern

Quelle: Erstellt mit Figma [14]



Abbildung 65: Erstellen eines neuen Aufgabeböckers

Quelle: Erstellt mit Figma [14]

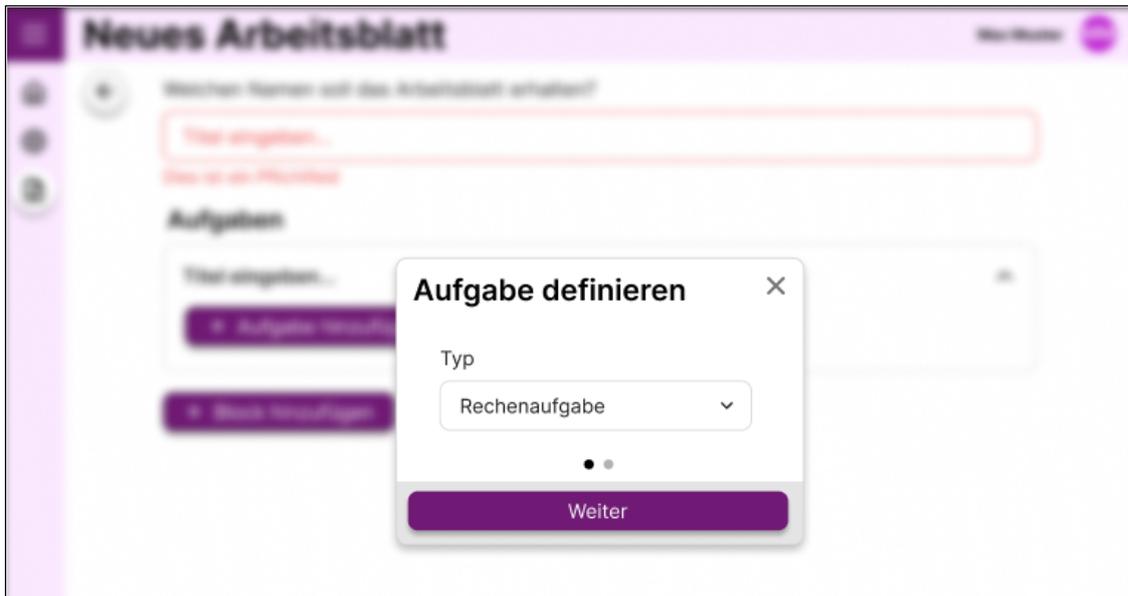


Abbildung 66: Auswählen von Aufgabentypen (Rechenaufgabe)

Quelle: Erstellt mit Figma [14]

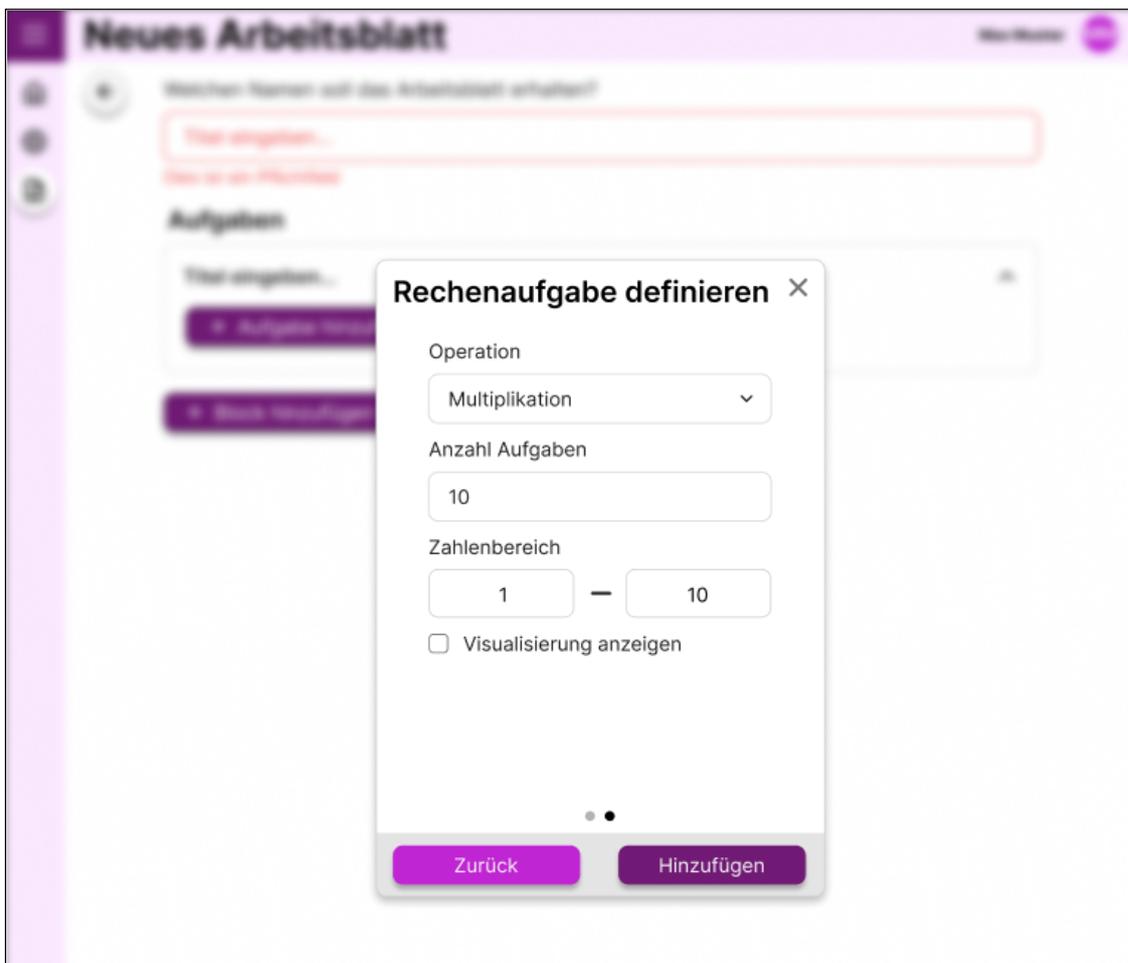


Abbildung 67: Eingabe von gewünschten Parametern für Rechenaufgabe

Quelle: Erstellt mit Figma [14]

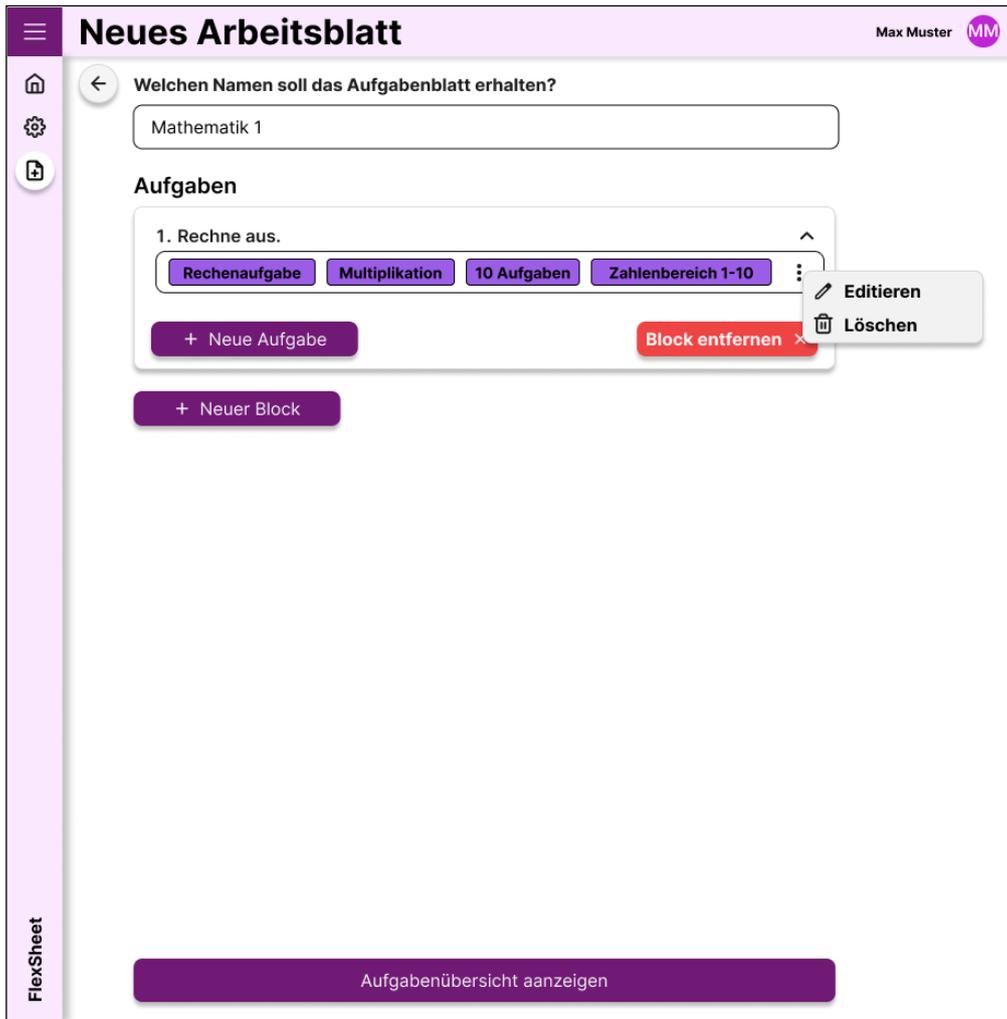


Abbildung 68: Anzeige neu erstellte Rechenaufgabe

Quelle: Erstellt mit Figma [14]

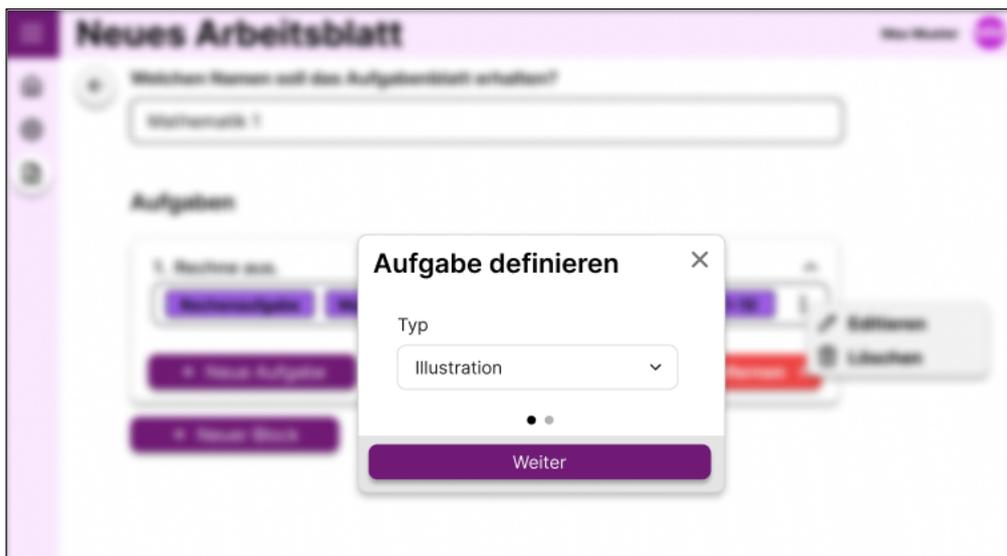


Abbildung 69: Auswählen von Aufgabentypen (Illustration)

Quelle: Erstellt mit Figma [14]

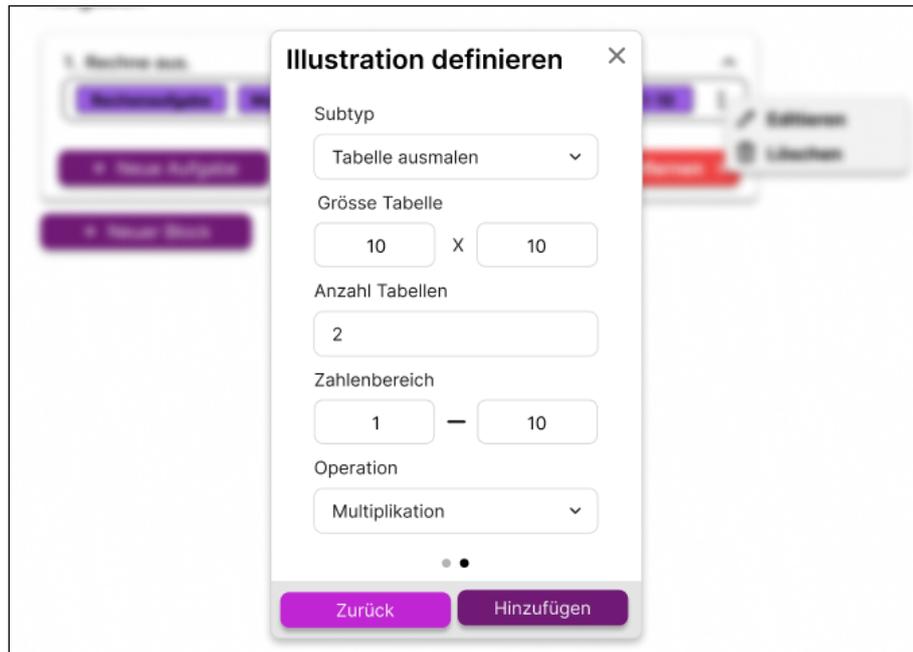


Abbildung 70: Eingabe von gewünschten Parametern für Illustration

Quelle: *Erstellt mit Figma [14]*

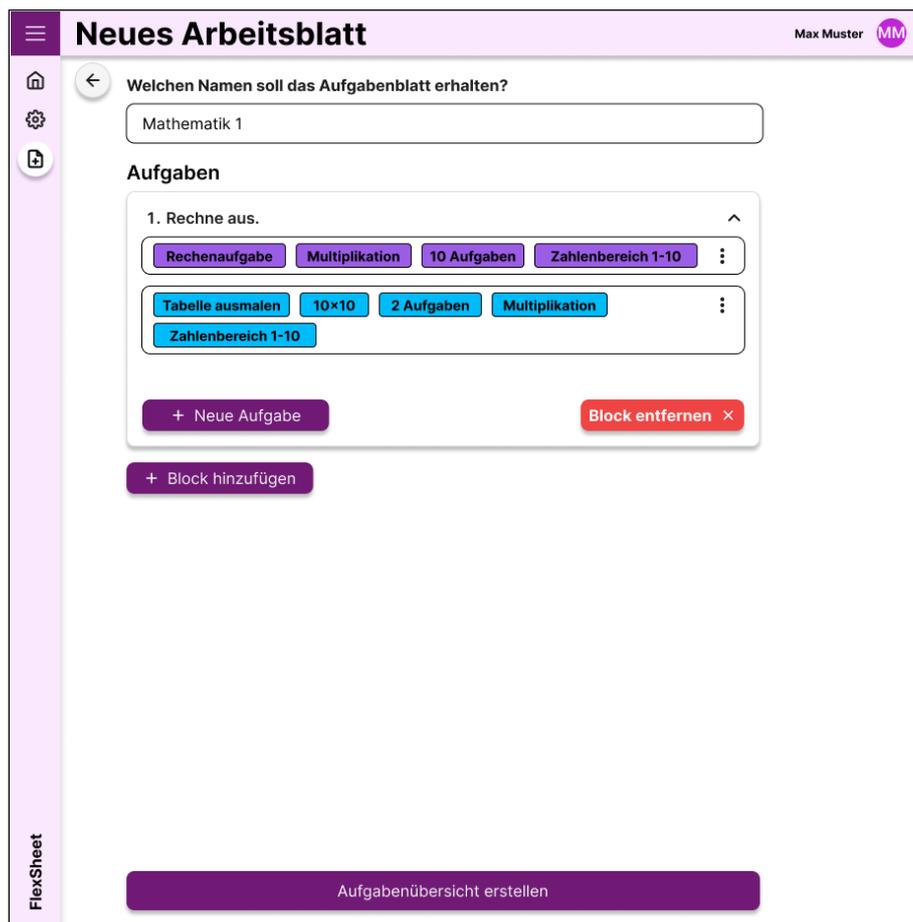


Abbildung 71: Anzeige neu erstellte Illustration

Quelle: *Erstellt mit Figma [14]*

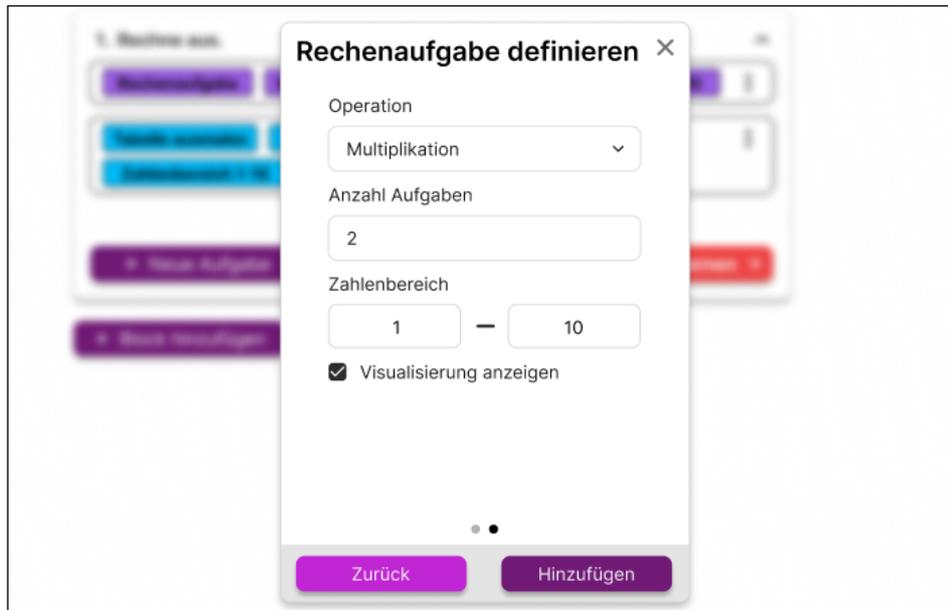


Abbildung 72: Erstellen von Rechenaufgabe mit Visualisierung

Quelle: Erstellt mit Figma [14]

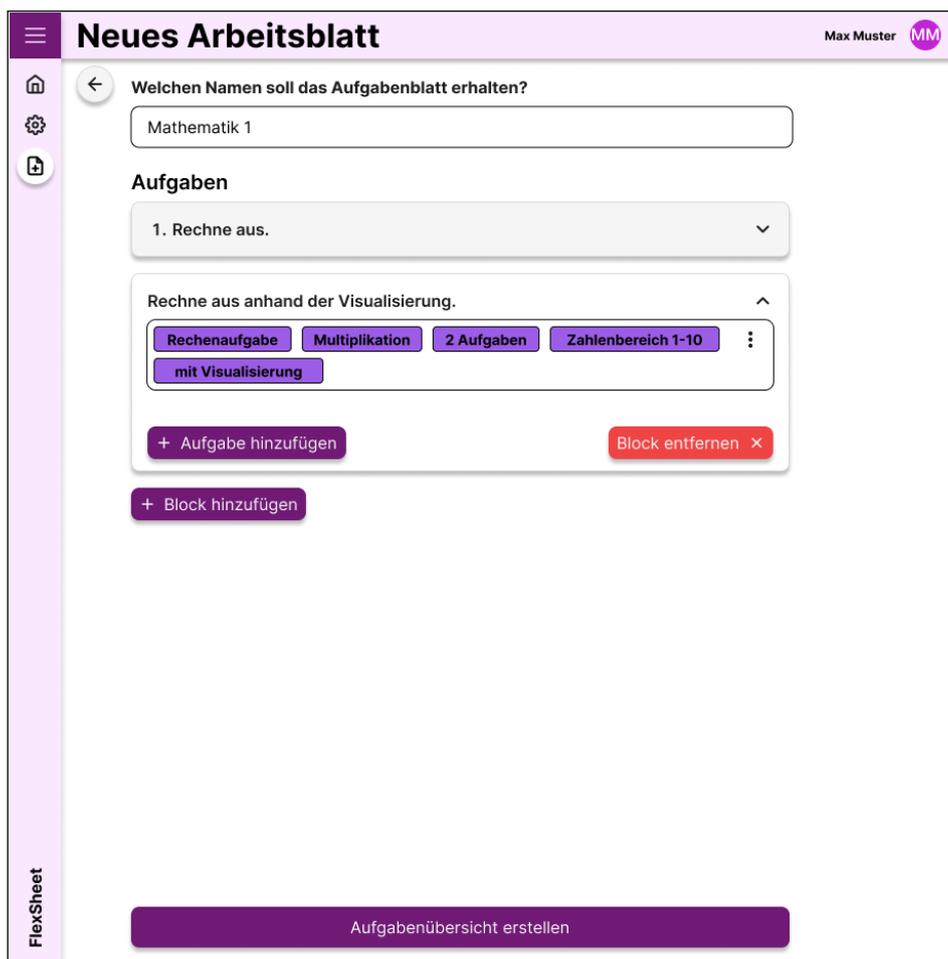


Abbildung 73: Anzeige neu erstellte Aufgabe mit Visualisierung

Quelle: Erstellt mit Figma [14]

FlexSheet

# Neues Arbeitsblatt

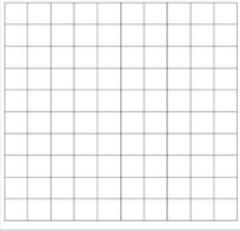
Max Muster 

Titel: Mathematik 1

1. Aufgabe: Block 1: Rechne aus.

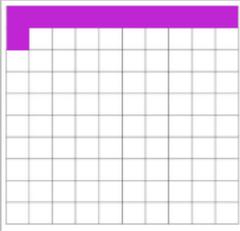
$5 * 3 = \underline{\quad}$   $3 * 6 = \underline{\quad}$   $7 * 5 = \underline{\quad}$   $8 * 9 = \underline{\quad}$   $2 * 4 = \underline{\quad}$   
 $1 * 3 = \underline{\quad}$   $6 * 4 = \underline{\quad}$   $8 * 7 = \underline{\quad}$   $2 * 8 = \underline{\quad}$   $6 * 4 = \underline{\quad}$

2. Aufgabe

$15 + 13 \rightarrow$  

$7 * 8 \rightarrow$  

Block 2: Rechne aus anhand der Visualisierung.

$6 + 5 = \underline{\quad}$  

$5 * 2 = \underline{\quad}$  

Speichern

Abbildung 74: Generierte Aufgaben in Übersicht anzeigen

Quelle: Erstellt mit Figma [14]

**Multiplikation - PDF Generieren** Max Muster MM

Vorlage  
Kinder mit Leseschwäche ▾

**Abstände**

Zwischen Blöcken ⓘ 1.5 cm  
Zwischen Zeilen ⓘ 1.5 cm

**Schrift**

Schriftgröße Titel 18 px  
Schriftfarbe Titel Blau

Schriftgröße Aufgaben 15 px  
Schriftfarbe Aufgaben Schwarz

Schriftart  
Arial

**Aufgaben**

Anzahl Aufgaben pro Block 10  
Maximale Anzahl Blöcke pro Seite 3

Generieren

Abbildung 75: Anwenden von Konfigurationen auf PDF

Quelle: Erstellt mit Figma [14]

**Konfigurationen** Max Muster MM

Name	
Kinder mit ADHS	✎ 📄 🗑️
Kinder mit Leseschwäche	✎ 📄 🗑️

Abbildung 76: Liste mit allen möglichen PDF Konfigurationen

Quelle: Erstellt mit Figma [14]

# Konfiguration editieren

Max Muster 



Name Konfiguration



**Abstände**

Zwischen Blöcken <sup>?</sup>

Zwischen Zeilen <sup>?</sup>

**Schrift**

Schriftgrösse Titel

Schriftfarbe Titel

Schriftgrösse Aufgaben

Schriftfarbe Aufgaben

Schriftart

**Aufgaben**

Anzahl Aufgaben pro Blatt

FlexSheet

 Speichern

Abbildung 77: Bearbeiten einer PDF Konfiguration

Quelle: *Erstellt mit Figma [14]*