

Adaptive Grid

IP5

ADHURIM GERVALLA, JULIAN FREY

Inhaltsverzeichnis

1	Abstract	3
2	Einleitung.....	3
2.1	<i>Was ist Adaptive-UI</i>	3
2.1.1	<i>Unterschied zwischen Adaptive-UI und Responsive-UI</i>	3
2.2	<i>Use-Case Enablerr Ausgangslage.....</i>	4
2.3	<i>Vision</i>	4
2.4	<i>Forschungsfragen</i>	4
2.5	<i>Methodik.....</i>	5
2.6	<i>Planung.....</i>	5
2.6.1	<i>Arbeitspakete.....</i>	5
2.7	<i>Aufbau</i>	7
3	Stand der Forschung	8
3.1	<i>Wissenschaftliche Arbeiten.....</i>	8
3.1.1	<i>Adaptive-UI in der Industrie</i>	15
3.2	<i>Zusammenfassung</i>	15
3.2.1	<i>Layout Integration.....</i>	16
3.2.2	<i>UI-Anpassungen durch AdaptML</i>	16
3.2.3	<i>Usability-Testing.....</i>	16
3.2.4	<i>Regeln basierte Entscheidungsfindung für Interface Adaption</i>	17
4	Adaptive Grid Konzept	18
4.1	<i>Adaptive Grid Aufbau.....</i>	18
4.1.1	<i>Prinzip.....</i>	18
4.1.2	<i>Aufbau und Variation</i>	19
4.1.3	<i>Intelligente Zellen.....</i>	21
4.1.4	<i>Ordnung</i>	22
4.2	<i>Regeln</i>	24
4.3	<i>Skalierbarkeit.....</i>	24
4.4	<i>Umgang mit Fehler</i>	24
4.5	<i>Performance</i>	25
5	Implementierung	25
5.1	<i>Ausgangslage.....</i>	25
5.2	<i>Technische Herausforderung</i>	27
5.2.1	<i>Universalität</i>	27
5.2.2	<i>Observers</i>	27
5.2.3	<i>Node-Pipelines</i>	28
5.2.4	<i>Zellenspeicher und ProxyCell Objekte.....</i>	28
5.2.5	<i>Navigation innerhalb des Adaptive Grids.....</i>	28
5.2.6	<i>CSS-Generierung</i>	29
5.2.7	<i>Berechnung der Zellen und der rowSpan Einträge</i>	30
5.2.8	<i>Abstand System.....</i>	31
5.2.9	<i>Konflikte mit externem CSS.....</i>	32
5.3	<i>Methodik.....</i>	32

5.4	<i>Technologien und Frameworks</i>	33
5.5	<i>Integration</i>	33
5.5.1	Svelte Grid ersetzen	34
5.5.2	Ungewollte Kinderelemente	35
5.6	<i>Ausfallverhalten implementieren</i>	35
5.7	<i>Integration der Regeln</i>	35
5.7.1	ruleSet	36
5.7.2	areaMap	36
5.7.3	Verarbeitungsprozess.....	36
5.7.4	Optimierung der Iteration	37
6	Evaluierung	41
6.1	<i>Vorgehensweise</i>	41
6.2	<i>Risiken in der Evaluierung</i>	41
6.3	<i>Zielgruppe analysieren</i>	41
6.4	<i>Bedienbarkeit des Enablerr</i>	42
6.4.1	Auswertung	42
6.4.2	Verbesserungsvorschläge.....	43
6.5	<i>Technische Umsetzung</i>	43
6.5.1	Verbesserungsvorschläge.....	43
6.6	<i>Integration als Universal Grid</i>	43
6.6.1	Auswertung	43
6.6.2	Verbesserungsvorschläge.....	43
6.7	<i>Evaluierung der Einsatzmöglichkeiten</i>	44
6.7.1	Enablerr Tasks	44
6.7.2	Automatisch generiertes UI	44
6.7.3	Dynamisches UI	44
7	Fazit	45
8	Literaturverzeichnis	46
9	Abbildungsverzeichnis	48
10	Tabellenverzeichnis	49

1 Abstract

In dieser Arbeit präsentieren wir eine Lösung für die Herausforderungen bei der Entwicklung von attraktiven und anpassbaren Webapplikationen. Unser Adaptive Grid ermöglicht die automatische Generierung eines User-Interfaces, das sich während der Laufzeit an die Anforderungen des Benutzers und des Systems anpasst. Wir untersuchen die Anwendung von Adaptive-UI-Prinzipien, um die Usability im automatisch generierten Frontend von Enablerr [1], einer revolutionären Business Solution, zu verbessern. Unsere Forschungsfragen beziehen sich auf die Auswahl von existierenden Konzepten und Technologien, die Anwendung von Neuordnungsregeln und die Behandlung von Fehlerfällen in unserer Lösung. Unser Adaptive Grid bietet ein solides und skalierbares Fundament, das ein grosses Potenzial für Ausbau- und Erweiterungsmöglichkeiten aufweist und sich in einem bisher unerforschten Bereich bewegt.

2 Einleitung

Eine grosse Herausforderung in der Entwicklung einer Webapplikation ist das attraktive Darstellen aller möglichen Features inklusive dessen Funktionen. Eine weitere Problematik stellt die attraktive Darstellung auf verschiedenen Bildschirmgrössen dar. Denn es existiert eine grosse Bildschirm Variabilität. Ebenfalls existieren immer mehr Webapplikationen, bei denen der Benutzer das UI selbst gestalten kann. Dies bedeutet, dass während der Laufzeit sich das UI immer wieder anpassen kann. All diese Herausforderung kombiniert führen zu grossen Aufwänden. Mit unserer Lösung, dem Adaptive Grid, möchten wir eine Lösung für diese Herausforderungen präsentieren. In den folgenden Unterkapitel erläutern wir, den Use-Case unseres Produktes und wie unser Adaptive Grid diese Probleme löst.

2.1 Was ist Adaptive-UI

Eine adaptive Benutzeroberfläche (UI) ist eine Art von Benutzeroberfläche, die sich automatisch an die Bedürfnisse und Eigenschaften des Benutzers anpasst. Dies kann beinhalten, die Grösse und Position der Elemente auf dem Bildschirm anzupassen, um besser für Benutzer mit bestimmten Fähigkeiten zugänglich zu sein, oder die Anzeige von Inhalten anzupassen, die für den Benutzer relevant sind. Adaptive-UI kann auch die Verwendung von verschiedenen Geräten und Bildschirmgrössen unterstützen und sich automatisch an die jeweiligen Einschränkungen anpassen.

2.1.1 Unterschied zwischen Adaptive-UI und Responsive-UI

Der Unterschied zwischen Adaptive-UI und Responsive-UI besteht darin, wie sie auf die Anforderungen des Benutzers reagieren. Responsive-UI passt sich an die Grösse des Bildschirms an, auf dem sie angezeigt wird. Sie nutzt flexible Layouts, Bilder und CSS-Medienabfragen, um sicherzustellen, dass die Inhalte auf unterschiedlichen Geräten und Bildschirmgrößen gut lesbar sind.

Ein Adaptive-UI geht über die blosser Anpassung an die Bildschirmgrösse hinaus und passt sich auch an die Bedürfnisse des Benutzers an. Dies kann zum Beispiel durch die Navigation, die Anzeige von Inhalten und die Interaktion mit der Anwendung an die Fähigkeiten des Benutzers anzupassen, umgesetzt werden. Adaptive-UI nutzt oft Informationen über den Benutzer, wie z.B. dessen Standort, Präferenzen und Verhaltensmuster, um die Benutzererfahrung zu verbessern.

2.2 Use-Case Enablerr Ausgangslage

Enablerr wird von den Herstellern als die revolutionäre Business Solution bezeichnet [1]. Die Softwarearchitektur ist radikal anders aufgebaut. "Enablerr unterscheidet sich deutlich von allem, was du heute kennst und bietet noch viel mehr: Erweiterbarkeit ohne Grenzen, Wartungsfrei und ohne lästige Update- oder Upgradeunterbrüche." [1] Enablerr befindet sich zurzeit noch in Entwicklung und ein grosser Teil des UI wird automatisch generiert abhängig von verschiedenen Parametern. Dies erschwert die zuverlässige Generation eines User-Interface, welches eine angenehme User-Experience ermöglicht. Wir möchten somit untersuchen, wie wir ein UI generieren können mithilfe von Adaptive User Interface Prinzipien, welche sich auf das Layout fokussieren.

2.3 Vision

Unser Ziel ist es ein System zu entwickeln, welches bei der automatischen Generation von Elementen im Frontend das UI stetig Usability freundlich konfiguriert. Während des Erarbeitens des Konzeptes möchten wir untersuchen, welche Adaptive-UI Prinzipien uns in diesem Projekt unterstützen können.

2.4 Forschungsfragen

- Welche existierenden Konzepte aus wissenschaftlichen Arbeiten oder aus der Industrie in Bezug auf Adaptive-UI können für unser Projekt verwendet werden?

Wie in vorgehenden Kapiteln beschrieben, existieren verschiedene wissenschaftliche Arbeiten bezüglich Adaptive-UI. Als erstes wird analysiert, welche existierenden Konzepte, Arbeiten und Aspekte bezüglich Adaptive-UI sich für die Implementierung in Enablerr eignen.

- In welchen Situationen soll die Neuordnung von Elementen ausgeführt werden?
 - o Existieren Ausnahmefällen?
 - o Existieren Elemente, welche immer der Neuordnung ausgeschlossen werden?
 - o Können bestimmte Komponenten eine Standardposition erhalten?
 - o Wie behandeln wir Fehlerfälle in unserer Lösung?
 - o Wie soll die automatische Neuordnung interagieren mit der manuellen Anordnung von Komponenten durch den Benutzer?

Durch die Beantwortung dieser Fragen soll definiert werden, in welchen Situationen unsere Lösung bestimmte Elemente verschieben darf und welche nicht. Wenn z.B. der Benutzer ein Element verschoben hat, stellt sich die Frage, ob dann dieses Element nicht mehr verschoben wird oder in welchen Fällen doch?

- Ist unsere Lösung universal anwendbar?
 - o Existieren Limitierungen?
 - o Welche Herausforderungen existieren bei der Implementation in ein bestehendes Projekt?

Wir möchten analysieren, ob unsere Lösung universell bei anderen Use-Cases angewendet werden kann. Dies beinhaltet sowohl die Analyse für weitere Komponenten im selben Projekt als auch in anderen Projekten.

2.5 Methodik

In einer ersten Phase werden wir uns auf die Recherche konzentrieren mit anschliessendem Fokus auf die Konzepterarbeitung. Da Adaptive-UI ein breites Thema ist, müssen wir untersuchen, welche Aspekte wir in unser Projekt übernehmen können. Anschliessend werden wir ein Konzept basierend auf diesen Recherchen erarbeiten. Im Anschluss werden wir die Implementation vornehmen. In der Implementierungsphase werden wir stetig evaluieren, ob die erarbeiteten Konzepte realistisch sind, welche Konzepte Erweiterungen benötigen und welche Konzepte nicht vorhersehbare Limitierungen aufweisen. Im Anschluss werden wir durch Live-Sessions die Lösung evaluieren lassen und anschliessend die Dokumentation mit der Evaluation ergänzen. Durch das Projekt durch werden wir regelmässig den Kontakt mit den Coaches suchen, sowie auch mit dem Enablerr Entwicklungsteam, um eine kohärente Zusammenarbeit zu gewährleisten.

2.6 Planung

Wir möchten Ihnen in diesem Abschnitt noch näherbringen, wie wir die Umsetzung der Arbeit geplant haben. Als erstes geben wir Ihnen deshalb eine einfache Übersicht und im Anschluss eine detailliertere Arbeitsaufteilung durch Arbeitspakete.

	1	2	3	4	5	6	7	8	9	10	PW	11	12	13	Frei	Frei	14	15
Projektarbeit	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
AP01 - Projektdefinition																		
AP02 - Recherche und Quellenfilterung																		
AP03 - Konzepterarbeitung																		
AP04 - Erste Implementation																		
AP05 - Implementation erweitern																		
AP06 - Evaluation																		
AP07 - Überarbeitung und Erforschung bzgl. Universalität																		

Abbildung 1: Übersicht Planung des Projektes

2.6.1 Arbeitspakete

2.6.1.1 AP01: Projektdefinition

Aufgabe: Definition der genauen Projektaufgabe mit dem Überthema "Adaptive User Interface"

Tätigkeiten:

- Recherche
 - o Sammlung von Quellen bzgl. Adaptive-UI
 - o Zusammenfassen besagter Quellen für die Beschreibung der Ausgangslage der Projektklärung
- Schreiben der Projektklärung (inkl. Korrekturen von Coaches)
- Table of Contents erfassen (inkl. Korrekturen von Coaches)

Abzuliefernde Ergebnisse:

- Projektklärung, welche von Coaches gelesen und akzeptiert wurde
- Table of Contents, welche von Coaches gelesen und akzeptiert wurde

2.6.1.2 AP02: Recherche und Quellenfilterung

Aufgabe: Weitere State of the Art Recherche und filtern, welche Quellen Aspekte haben, welche für unser Thema relevant ist.

Tätigkeiten:

- Recherche
 - o Weitere Sammlung von Quellen bzgl. Adaptive-UI
 - o Dokumentieren der Quellen
- Übernahme abgesprochener Stellen von Projektklärung in wissenschaftliches Dokument

- Erste Version der Einleitung fertig

Abzuliefernde Ergebnisse:

- Kapitel Einleitung
- Kapitel Stand der Forschung

2.6.1.3 AP03: Konzepterarbeitung

Aufgabe: Erarbeitung des Grundkonzeptes der Lösung.

Tätigkeiten:

- Genauere Analyse der Enablerr Plattform
- Unsere Lösung im wissenschaftlichen Dokument beschreiben
- Erarbeiten eines Lösungskonzeptes

Abzuliefernde Ergebnisse:

- Kapitel Lösungskonzept

2.6.1.4 AP04: Erste Implementation des Basic Lösungskonzeptes

Aufgabe: Das Lösungskonzept (Basic Version), welches im AP03 definiert worden ist, soll nun im Enablerr Projekt implementiert werden

Tätigkeiten:

- Erste Implementation des Lösungskonzept (Basic) in Enablerr

Abzuliefernde Ergebnisse:

- Erstes Lösungskonzept implementiert
- Implementation dokumentiert

*2.6.1.5 AP05: Implementation erweitern***Tätigkeiten:**

- Fortlaufende Erweiterung der Lösung
- Implementation fortlaufend dokumentieren
- Konzeptkapitel überarbeiten wo notwendig

Abzuliefernde Ergebnisse:

- Implementierte Lösung
- Implementationen dokumentiert
- Kapitel Implementation

*2.6.1.6 AP06: Evaluation***Tätigkeiten:**

- Implementation evaluieren durch Live-Testing und Fragebögen
- Abstract schreiben

Abzuliefernde Ergebnisse:

- Ausgefüllte Fragebögen
- Testing dokumentiert
- Kapitel Evaluation
- Kapitel Abstract

2.6.1.7 AP07: Überarbeitung und Erforschung bzgl. Universalität

Aufgabe: Implementation analysieren in Bezug auf Universalität → Wie einfach kann diese Implementation in ein anderes Projekt implementiert werden?

Tätigkeiten:

- Analyse dokumentieren

- Feedback von Coaches in Dokumentation umsetzen
- Dokumentation abgeben

Abzuliefernde Ergebnisse

- Analyse Universalität
- Finale Version der Dokumentation abgegeben

2.6.1.8 AP08: Erstellung Webseite

Aufgabe: Es ist der Wunsch der FHNW, dass gemäss einer vorgegebenen Anleitung eine Webseite des Projektes erstellt werden soll.

Tätigkeiten:

- Webseite mit Text erstellen
- Passende Bilder finden und einfügen

Abzuliefernde Ergebnisse:

- Finale Webseite an vorgegebenem Ort abgelegt

2.6.1.9 AP09: Erstellung Präsentation

Aufgabe: Die Arbeit soll durch eine Präsentation vorgetragen werden

Tätigkeiten:

- Powerpoint Präsentation erstellen
- Präsentation vorgetragen

Abzuliefernde Ergebnisse:

- Powerpoint Präsentation

2.7 Aufbau

Anschliessend dieser Einführung in unser Projekt folgt in Kapitel 3 unsere Recherche bezüglich Adaptive-UI. Wir werden in diesem Kapitel unsere Resultate aufzeigen, was existiert, sowohl in der Wissenschaft als auch in der Industrie. Ebenfalls werden wir dort aufzeigen, welche Arbeiten und/oder Produkte für unser Projekt relevant sein können. Anschliessend folgt Kapitel 4. Hier werden wir das Konzept von unserer Lösung genauer beschreiben. Hier werden wir erklären, wie wir die Architektur und die konzeptionelle Lösung genauer beschreiben. Daraufhin folgt Kapitel 5, in dem wir genauer beschreiben, wie wir unser Konzept in die Enablerr Plattform integrieren. Ebenfalls stellen wir genauer vor, welche Herausforderungen wir antreffen und wie wir diese lösen. Kapitel 6 werden wir unser implementiertes Produkt evaluieren durch Fragebögen und Live-Testing sessions. Anschliessend dem Testing wir unsere Resultate darlegen und unsere Schlussfolgerungen definieren.

3 Stand der Forschung

In diesem Abschnitt untersuchen wir die Wissenssättigung im Bereich der Adaptive User Interfaces.

3.1 Wissenschaftliche Arbeiten

Table 1: Wissenschaftliche Recherche Adaptives UI

Titel	Jahr	Venue	Zusammenfassung	Erwähnte Technologien	Schlüsselfazit
AUIT - The Adaptive User Interfaces Toolkit for Designing XR Applications [2]	2022	Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology	Adaptive User Interfaces können Extended Reality (XR) Applikationen verbessern, indem Sie sich an den Kontext des Benutzers anpassen. Bei XR-Applikationen müssen noch mehr Faktoren beachtet werden, da sich die Applikationen im 3D Raum befinden, und nicht im 2D Raum. Dies soll mit diesem Toolkit vereinfacht werden.	AUIT, VR, AR, XR, rule-based adaptation	Konzentriert sich auf XR Use-Case. Jedoch diskutiert es mehrmals, wie es in der 2D Welt umgesetzt ist und ob dies adaptiert werden kann auf 3D.
A Contextual Framework for Adaptive User Interfaces: Modelling the Interaction Environment [3]	2022	University of Luxembourg	Diese Arbeit befasst sich mit dem Ansatz, dass das UI abhängig von der Umgebung modelliert werden soll. Es beschreibt, dass in vielen anderen Arbeiten der User- und Systemkontext benützt wurde, jedoch wurde die Umgebung nicht gross in Betracht gezogen. Sie schlagen ein Framework vor, in welchem der Benutzer-, System- und Umgebungskontext in Betracht gezogen wird.		Konzentriert sich mehr auf die verschiedenen Kontexte und behandelt das Layout nicht wirklich.
Semantic-Based Design of an Adaptive-UI [4]	2022	Communications in Computer and Information Science book series (CCIS, volume 1625)	In dieser Arbeit wird ein Konzept vorgestellt, in welchem die bestehenden Ansätze (Model-driven, ontologisch aufgebaut) in Betracht gezogen werden und ein ontologischer Ansatz verfolgt wird, mit dem semantische Modelle entstehen, welche auf der OSTIS Technologie basieren.	OSTIS, Ontologischer UI-Ansatz, Model-basierend	Löst einige Probleme in Bezug auf Benutzer- und Systemkontext. Das Layout bzw. die Generierung des Layouts und dessen Anordnung wird jedoch nicht erwähnt.

Towards a Model-Driven Ontology-Based Architecture for Generating Adaptive User Interfaces [5]	2022	Lecture Notes in Network and Systems book series (LNNS, volume 483)	Hier werden die bestehenden model-driven und Ontologie basierten Ansätze genommen und erweitert durch eine Erstellung eines semantischen Modelles und der Wiederbenützung von verschiedenem Wissen.	Onthologischer UI-Ansatz, Model-basierendes UI	Erweitert die bestehenden Adaptive-UIs, jedoch wird das Layout nicht beschrieben.
Automatic generation of user-sensitive and application-sensitive self-adapted UI system for smartphone applications [6]	2022	2022 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)	Diese Arbeit fokussiert sich auf Adaptive-UIs von Smartphones. Sie proponieren in der Arbeit ein Framework (AGUSASSSS), um die Applikationen natürlicher und effizienter gestalten zu können.		In einer simplen Applikation hat Ihr adaptives UI einige Vorteile. Jedoch bei komplexeren Applikationen weist es einige Lücken auf. Die Arbeit beschäftigt sich jedoch ebenfalls mit Layouts, was für unsere Arbeit relevant sein könnte.
Adaptive user interfaces and universal usability through plasticity of user interface design [7]	2021	Computer Science Review Volume 40, May 2021, 100363	Studie über die universelle Usability, Plastizität von User Interfaces und Erleichterung von Interface Entwicklung mit universeller Usability mit dem Ziel die günstigste Richtung für zukünftige Adaptive User Interface Designs zu finden. Sie basiert auf 165 Papers im Zeitraum von 55 Jahren.		In der Studie werden hauptsächlich folgende 3 Bereiche untersucht: Künstliche Intelligenz, User Modeling und Mensch-Computer Interaction. Es geht zwar auch um das Generieren von User Interfaces, allerdings nicht in dynamischer Form. Die Generierung erfolgte durch das Füttern eines Algorithmus mit Daten.
ARtention: A design space for gaze-adaptive user interfaces in augmented reality [8]	2021	Computers & Graphics Volume 95, Science Direct, 2021	In diesem Paper wird ARtention vorgestellt: Ein Konstruktionsraum für die Blickwechselwirkung, die speziell auf In-situ-AR-Informationsschnittstellen zugeschnitten ist. Dabei geht es hauptsächlich um das Zusammenspiel von Augmented Reality und der Adaptierung des Interfaces basierend auf den Blickwinkel Informationen, die gesammelt werden.	Augmented Reality	Sicher interessante Punkte, allerdings verfehlt das Thema unseren Themenbereich.

Adapting User Interfaces with Model-based Reinforcement Learning [9]	2021	Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems	Diese Arbeit behandelt die Thematik, möglichst positive User Experience zu erzeugen, indem Überraschungen oder Relearning Anstrengungen verhindert werden. Diese negativen Effekte sind schwer vorherzusagen, da diese von gewissen Faktoren anhängig sind. Die Faktoren sind flexibel und entwickeln sich mit der Zeit. Sie versuchen dies zu lösen, indem sie vorteilhafte Interface-Wechsel von nicht so vorteilhaften unterscheiden können. Das machen sie mit Hilfe einer Model-basierten Lernmethode, welche Sequenzen von Adaptionen anschauen und versucht deren Effekt auf das Interface vorherzusagen.	Python für das Model, TensorFlow für das neurale Netzwerk	Ein Interessanter Ansatz, indem es hauptsächlich darum geht eine Lernmethode zu trainieren, die Sequenzen von Adaptionen vorhersagen kann. Diese Methodik können wir allerdings in unserer Arbeit nicht anwenden, da wir keinen Machine Learning Ansatz verwenden.
Web Map Effectiveness in the Responsive Context of the Graphical User Interface [10]	2021	ISPRS Int. J. Geo-Inf., EISSN 2220-9964, Published by MDPI	Diese Arbeit untersucht die Effektivität eines Karten-Interfaces, dass für Desktop und Mobile Geräte entwickelt wurde, zu bestimmen. Dabei werden im Wesentlichen drei Regeln untersucht: Design von kleinen zu grossen Bildschirmen, Design von grossen zu kleinen Bildschirmen und 2 unterschiedliche Designs. Die Effektivität des Interfaces wurde an 120 Teilnehmern mit Eye-Tracking gemessen.	Eye Tracking Software, JavaScript, HTML, CSS, SVG	Diese Arbeit stellt eine konkrete Untersuchung einer Kartenansicht wie Google Maps dar, dass eine Limitierte Anzahl an Funktionen bietet. Dies entspricht leider nicht unserer Vorstellung eines Adaptive Grids und stellt daher für uns keine Hilfe dar.
Integrated model-driven development of self-adaptive user interfaces [11]	2020	Software and Systems Modeling vol 19, pages 1057 - 1081	In dieser Arbeit werden zwei Sprachen entwickelt, welche ein Adaptive-UI generieren können in Runtime. Einerseits wurde eine Sprache erstellt, welche den Benutzerkontext definiert (ContextML), andererseits wurde eine Sprache erstellt, welche sich auf die UI-Anpassungen bezieht (AdaptML)	ContextML, AdaptML, Model based UI, IFML	Die Arbeit bezieht sich auf zwei einzelne Dinge: UI-Anpassung und Benutzerkontext. Die UI-Anpassungen könnte für unser Projekt sich als interessant erweisen.
Model-driven engineering and usability evaluation of self-adaptive user interfaces [12]	2020	ACM SIGWEB Newsletter, Issue Autumn 2020	In dieser Doktorarbeit wird eine model-driven UI-Lösung implementiert. Aus dieser Lösung entstanden ebenfalls verschiedene Arbeiten in der Industrie. Es wird ebenfalls über Adaptive-UIs in AR und VR gesprochen, jedoch wurde hierzu keine Implementation durchgeführt. Des Weiteren führt es eine Sprache ein für das UI eigentlich zu adaptieren und um die Usability zu testen.	Model based UI, rule-based execution engine	Der benutzerkontextbezogene Teil der model-based UI ist für unser Thema weniger interessant. Jedoch könnte die UI für das Formen der Webseite und das Usability-Testing interessant sein.

The Intelligent Medical Platform: A Novel Dialogue-Based Platform for Health-Care Services [13]	2020	Computer, Volume 53, Issue: 2, February 2020	Hier entwickelte man eine Online-Plattform, in welcher man medizinische Hilfestellung und Diagnosen erhalten kann. Diese Online-Plattform wurde umgesetzt in einem Adaptive-UI mit einem model-based Prinzip.	Model based UI	Grosser Teil bezieht sich auf den gesundheitlichen Aspekt. Der andere Teil fokussiert sich auf den Benutzerkontext. Somit scheinen keine Korrelation vorzuliegen zwischen dieser und unserer Arbeit.
Engineering Privacy-aware Smart Home Environments [14]	2020	Proceedings of the 12th ACM SIGCHI Symposium on Engineering Interactive Computing Systems	Diese Arbeit erstellte ein Framework, in welchem die Privatsphäre von den Benutzern bewahrt, wird durch ein Adaptive-UI. Für die Umsetzung wurde mit einem Model basierten Ansatz umgesetzt, welches mit einem Machine-Learning Algorithmus unterstützt wird. Des Weiteren wird die "Rej Policy Language" verwendet für die Regelung der Privatsphäre. Dieses Gesamtpaket wurde noch addiert mit einem "Virtual User Modelling and Simulation Cluster" (VUMS Cluster), welche die Benutzermöglichkeiten und UI Einstellungen modelliert.	Model based UI, Rej Policy Language, Vums Cluster	Model-based UI bezieht sich auf Benutzerkontext. Der Machine Learning Ansatz ist interessant, jedoch ebenfalls für unseren Ansatz nicht anwendbar. Die Rej Policy Language bezieht sich auf die Regelung von Privatsphäre, was in unserer Arbeit nicht von Relevanz ist. Der Vums Cluster könnte ein möglicher Ansatz sein für die Implementierung der UI-Einstellungen.
Component-based development of adaptive user interfaces [15]	2019	EICS '19: Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems	In dieser Arbeit geht es, wie beim Model basierten Ansatz darum, die Benutzerfreundlichkeit von Adaptive User Interfaces zu verbessern, in dem auf Kontextveränderungen zur Laufzeit reagiert wird. Im Gegensatz zu den Modelgetriebenen Ansätzen, die in Sachen Flexibilität, Wiederverwendbarkeit und Kompatibilität mit modernen JavaScript Frameworks wie Angular versagen, wird in dieser Arbeit ein Komponenten basierter Ansatz untersucht, dass sie CoBAUI nennen.	JavaScript, Angular	Im Decision Making Abschnitt wird erläutert, wie das UI entscheidet, ob Anpassungen am UI vorgenommen werden sollen oder nicht. Die Entscheidungen werden basierend auf Regeln gefällt, welche möglichst global definiert werden. Dieser Ansatz ist für unsere Arbeit sehr interessant, da unser Adaptive Grid ebenfalls auf Regeln basierend agieren wird.
A Framework for Adaptive User Interface Generation based on User Behavioural Patterns [16]	2019	2019 Moratuwa Engineering Research Conference (MERCon)	In diesem Paper wird beschrieben, dass die meisten Arbeiten nur Konzeptlösungen für ein optimales Adaptive User Interface System anbieten und auf produktiver Ebene für Mainstream Applikationen keine Anwendung finden. Als Lösung für dieses Problem, präsentieren sie eine generische Softwareplattform für die automatische Generierung von Adaptive User Interfaces, indem Benutzerverhaltensmuster analysiert und Web-Benutzeroberflächen mithilfe von maschinellem Lernen angepasst werden.	NodeJS, MongoDB, JavaScript, PostgreSQL, Django, Python	Der Fokus dieser Arbeit liegt in der Adaptierung von User Interfaces basierend auf Userverhalten. Dies wird mit einem Machine Learning Ansatz gemacht, was wiederum nicht in unserem Scope ist.

Adaptive User Interface of Learning Management Systems for Education 4.0: A Research Perspective [17]	2019	Journal of Physics: Conference Series, Volume 1235	Der Adaptive User Interface Ansatz wird hier im Zusammenhang mit Lernplattformen für Schüler untersucht. Dabei werden Lösungen von anderen Forschungsergebnissen untersucht, verglichen und bewertet.		In diesem Paper konnten wir keine für uns relevanten Informationen extrahieren, da es hier explizit um Lernplattformen geht und nicht wirklich ein Ansatz für Regeln basiertes Adaptive User Interface beschrieben wird.
Rule based adaptive user interface for adaptive E-learning system [18]	2019	Educ Inf Technol 24	Forschung eines generischen Ansatzes, um die Lerninhalte mit Adaptive User Interface Komponenten bereitzustellen, die auf den Lernstilen der Lernenden basieren. Der für die Arbeit gewählte Lernstil ist das Felder-Silverman Learning Style Model (FSLSM). Der vorgeschlagene Ansatz definiert generische Regeln, die automatisch für jeden Online-Kurs mit adaptiven Inhalten generiert werden.	Felder-Silverman Learning Style	Als Lösungsvorschlag wird ein Regel basierter Ansatz definiert. Die Regeln bilden die Basis für Generierung des Designs einzelner Komponenten. Ebenfalls wird eine Concept Map beschrieben, die den Lernenden den optimalen Lernpfad vorschlagen sollte.
Gaze-based Product Filtering: A System for Creating Adaptive User Interfaces to Personalize Stateless Point-of-Sale Machines [19]	2019	UIST '19: The Adjunct Publication of the 32nd Annual ACM Symposium on User Interface Software and Technology	In dieser Arbeit wird ein Ansatz beschrieben, der basierend auf Blickinformationen des Users das Interface automatisch anpasst. Dabei werden Interface-Elemente die als nicht relevant angesehen werden, aus dem Sichtfeld entfernt und Elemente die als relevant angesehen werden, ins Zentrum geschoben.	Gaze based AUI	Diesem Sheet können wir leider keinerlei für uns relevante Daten entnehmen, da wir nicht mit gaze based Daten arbeiten.
Model-based adaptive user interface based on context and user experience evaluation [20]	2018	Journal ohn Multimodal User Interfaces 12, p. 1 - 16	In dieser Arbeit wird ein Model basierendes UI vorgeschlagen, welches Domänen- und geräteunabhängig ist. Das System ist abhängig vom Benutzerkontext und der User Experience.	Model based UI	Hier konzentriert man sich sehr fest auf den Benutzerkontext, dessen Domäne und Geräteigenschaften. Währenddessen die Adaptionregeln, für die Generierung des UI, schon angewendet werden, sind diese noch nicht in einem sehr stark entwickelten Format. Es wird im Fazit erwähnt, dass als nächstes diese Regeln ausgebaut werden soll in Bezug auf die User Experience. Deshalb scheinen Sie das Thema, welches für uns relevant ist, noch nicht spezifisch bearbeitet zu haben.
Adaptive User Interface for a Personalized Mobile Banking App [21]	2018	UMAP '18: Adjunct Publication of the 26th Conference on User Modeling, Adaptation and Personalization	Hier wird ein System für Mobile Banking entwickelt. PERMA soll ein adaptives System sein, in welchem personalisierter Inhalt, Präsentation und Navigationshilfe angeboten wird.	Adaptive-UI durch User Action Prediction	Diese Arbeit bezieht sich auf die Action Prediction und dessen Algorithmus. Die Generierung des Layouts, bzw. des UI in Bezug auf Usability wird nicht genauer diskutiert.

Object-oriented User Interface Customization Framework: Customizing Complex User Interfaces to Improve Usability and User Performance [22]	2018	Purdue University	Hier wird das Object-oriented User Interface Customization Framework auf drei Use-Cases getestet. Zwei dieser Use-Cases beziehen sich auf Adaptive-UI, das letzte auf Adaptable UI. Die Use-Cases können gelöst werden und die Arbeit erwähnt, dass eine Kombination von Adaptive-UI und adaptable UI durchaus attraktiv sein kann.	Adaptable UI, OOUIC	Diese Arbeit bezieht sich viel auf Benutzerkontext. Jedoch bezieht sich der einte Teil auf die Drag & Drop Funktionalität, was bei unserem Projekt ebenfalls auftritt.
Adaptive user interface optimization for multi-screen based on machine learning [23]	2018	2018 IEEE 22nd International Conference on Computer Supported Cooperative Work in Design (CSCWD)	In der heutigen IT besteht eine grosse Anzahl an verschiedenen Bildschirmgrößen. Für ein UI wird es immer schwieriger, ein attraktives UI für all die verschiedenen Use-Cases anzubieten. In dieser Arbeit wird ein Intelligentes UI vorgeschlagen, dessen Layout sich automatisch an das Geräteprofil anpasst.	Intelligent UI, Machine Learning	In dieser Arbeit wird öfters von der Transformation des Layouts gesprochen, welches Interessant für unser Projekt ist. Jedoch bezieht sich die Arbeit schnell auf den Aspekt des Machine-Learnings und den Transformation-Algorithmus.
Deep Sequential Recommendation for Personalized Adaptive User Interfaces [24]	2017	IUI '17: Proceedings of the 22nd International Conference on Intelligent User Interfaces	Adaptive User Interfaces sollen kontextabhängige Änderungen in Echtzeit anzeigen und das User Interface dementsprechend ändern. Als Problem wird beschrieben, dass dies schwierig ist, wenn nur wenig Daten pro User vorhanden sind. Mit einem Deep Learning Ansatz wird versucht, dieses Problem zu lösen, indem Benutzerinteraktionsmuster effizient gelernt werden, und kollaborative Filtertechniken eingesetzt werden, die den Austausch von Daten zwischen Benutzern ermöglichen.	Deep Learning, Machine Learning	Diese Arbeit behandelt einen Deep Learning Ansatz was nicht im Scope unserer Arbeit ist.
The Influence of Personality Traits and Cognitive Load on the Use of Adaptive User Interfaces [25]	2017	IUI '17: Proceedings of the 22nd International Conference on Intelligent User Interfaces	Diese Studie zeigt systematische und individuelle Unterschiede in der Nutzung der adaptiven Funktionen, die mit den stabilen Benutzermerkmalen «Need for Cognition and Extraversion» korrelieren.		Diese Studie behandelt die Messung von Performance relevanten Daten.

Model-Driven Context Management for Self-adaptive User Interfaces [26]	2017	Part of the Lecture Notes in Computer Science book series (LNISA, volume 10586)	Um die komplexe und umständliche Aufgabe des Kontextmanagements zu bewältigen, stellen sie einen modellgetriebenen Ansatz zur Entwicklung eines Kontextmanagers vor, der adaptive User Interfaces unterstützt. Dieser Ansatz besteht aus einer Kontextmodellierungssprache namens ContextML zur Spezifikation verschiedener Situationen des Nutzungskontexts. Basierend auf dem spezifizierten Kontextmodell ermöglicht dieser Ansatz die automatische Generierung von Kontextdiensten zur Überwachung von Nutzungskontextparametern.	Angular, ContextML	In dieser Arbeit wird ein Kontext Management System für Sensoren und heterogenen Quellen. In unserer Arbeit befassen wir uns allerdings nicht solchen Quellen. Daher ist diese Arbeit für uns nicht relevant.
An Adaptive User Interface for an E-learning System by Accommodating Learning Style and Initial Knowledge [27]	2017	ICTVT 2017	In dieser Arbeit geht es darum eine Strategie zu entwickeln, die eine individualisierte Lernumgebung schafft, welche den Ansprüchen der Schüler entspricht. Sie nennen dies ein Adaptives E-Learning System. Das Design des E-Learning System passt sich an verschiedene Lernstile und Wissensniveaus an.	Felder-Silverman Learning Style	Obwohl es hier um eine E-Learning Plattform geht, wird diesmal kein Machine Learning Ansatz verwendet. Stattdessen wird ein Set von Regeln definiert, anhand dessen das Design der Interfaces generiert wird. Diese Methodik ähnelt unserem Vorhaben.
Design Space Exploration of Adaptive User Interfaces [28]	2017	Orange Labs, 28 chemin du Vieux Chêne, F-38240 Meylan (France)	In diesem Paper werden verschiedene Methoden verglichen, wie man im laufenden Entwicklungsprozess adaptive Interfaces implementieren kann, welche sich für die Situation am besten eignen.		In diesem Paper befinden sich keinerlei Informationen, die für uns relevant sein könnten.

1.1 Adaptive-UI in der Industrie

Nach einer gründlichen Recherche konnten wir keine spezifischen Produkte, Frameworks oder Sprachen finden, die Adaptive-UI anbieten. Wir stellten fest, dass in der Branche der Begriff "Adaptive-UI" oft synonym verwendet wird mit "Responsive UI" [29] [30].. Unsere Untersuchung ergab, dass der Begriff "Adaptive-UI" häufig für Marketingzwecke verwendet wird und nicht der wissenschaftlichen Definition entspricht.

3.2 Zusammenfassung

In unserer Recherchen-Phase haben wir insgesamt 27 Papers untersucht. Um möglichst viele Phasen des Technologischen Fortschritts abzudecken, haben wir die Recherche über eine Zeitspanne von fünf Jahren aufgeteilt und für jedes Jahr jeweils vier bis fünf Papers untersucht. Die untersuchten Arbeiten behandeln alle das Thema Adaptive User Interfaces.

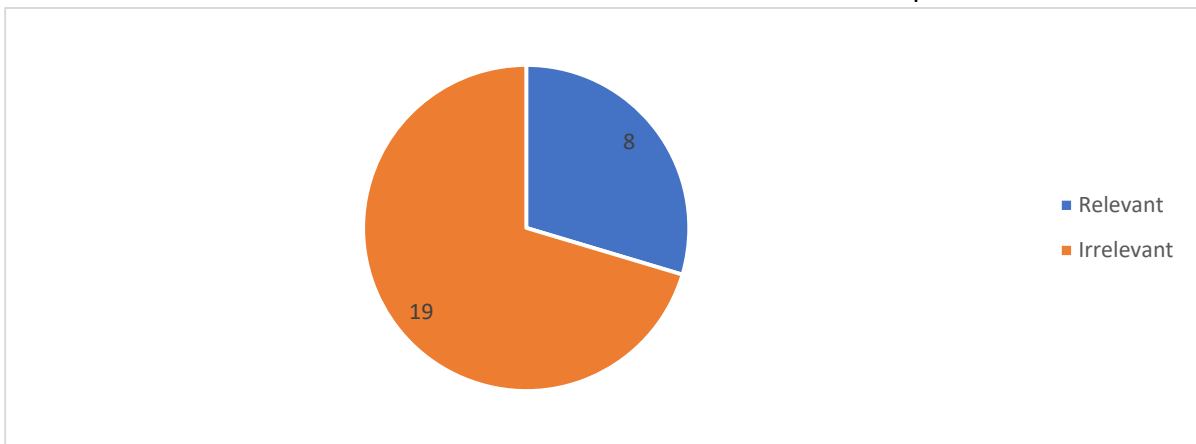


Abbildung 2: Filterung der wissenschaftlichen Recherche – Relevante und irrelevante Papers

In Abbildung 2 kann man erkennen, dass wir gesamthaft acht Arbeiten als relevant markiert. Die Arbeiten, welche wir als irrelevant markiert haben, fokussieren sich nicht auf Aspekte von Adaptive-UI, welche wir in unserem Projekt behandeln möchten. In vielen Fällen behandeln Sie den Benutzerkontext und fokussieren sich nicht konkret auf die Generierung des Layouts.

Des Weiteren haben wir eine Analyse der Technologien durchgeführt, welche in Abbildung 3 ersichtlich ist.

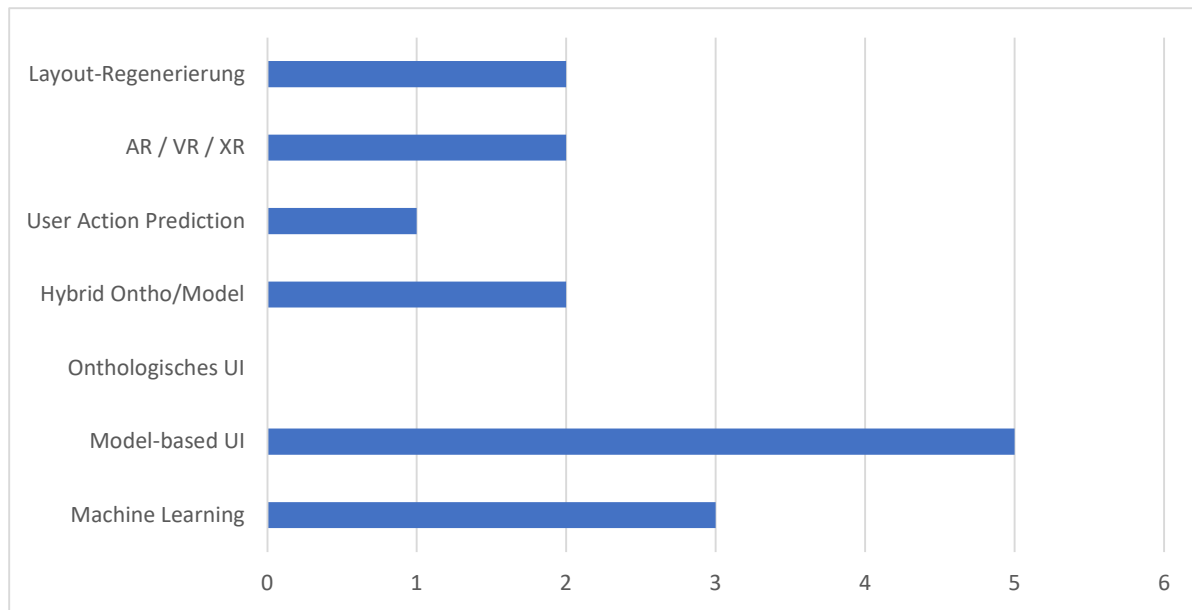


Abbildung 3: Erwähnte Technologien

3.2.1 Layout Integration

Bei einem Adaptive User Interface spielt das Layout eine zentrale Rolle, da sie das Grundraster des Interfaces definiert, indem verschiedene Elemente platziert werden können [6]. Der initiale Zustand ist ein vorgefertigtes Layout, welche keinerlei benutzerspezifischen Adaptationen enthalten. In einem zweiten Schritt wird aus diesem Layout ein Benutzer spezifisches Layout gefertigt. Dabei unterscheiden Sie 3 wesentliche Schritte:

1. Informationen über den Benutzer sammeln
2. Generierung von Bewegungsregeln
3. Generierung des finalen User Interfaces

Die drei Schritte sind aufbauend. Die Bewegungsregeln werden basierend auf den Benutzerinformationen definiert. Sie definieren, wie sich die Elemente auf dem Interface zu bewegen haben. In der dritten Phase werden die Regeln angewendet und das Layout wird optimiert. Spezifisch das Konzept der Bewegungsregeln könnte ebenfalls in unser Projekt übernommen werden.

3.2.2 UI-Anpassungen durch AdaptML

In der Arbeit "Integrated model-driven development of self-adaptive user interfaces" haben wir AdaptML genauer untersucht. AdaptML ist eine Modellierungssprache, welche erstellt wurde, um User Interface Anpassungsregeln vom Kontext Model trennen zu können. AdaptML sollte ebenfalls möglichst flexibel definierbar sein, um verschiedenste Adaptionsregeln erstellen zu können. Für eine gut gegliederte Struktur der Regeln wurde hier eine Modellierungssprache erstellt. Während der Analyse dieser Sprache stellten wir fest, dass sie eine starke Koppelung mit der Sprache ContextML aufweist, welche ebenfalls in derselben Arbeit entwickelt wurde und sich auf den Benutzerkontext bezieht. Deshalb haben wir uns gegen eine Integration von AdaptML in unserem Projekt entschieden. Jedoch können gewisse Modellierungskonzepte und weitere Prinzipien für unser Projekt hilfreich sein.

3.2.3 Usability-Testing

In der Arbeit "Model-driven engineering and usability evaluation of self-adaptive user interfaces" wird ein spannender Ansatz für das Usability-Testing beschrieben. Während die Endbenutzer während der Nutzungszeit mit dem interaktiven System interagieren, ermöglicht das System die Bewertung der Endbenutzerzufriedenheit durch die Kombination

von Kontextüberwachung mit der Sammlung von sofortigem Benutzerfeedback. Die Implementation von automatischem Usability-Testing können wir aus Zeitgründen nicht durchführen, jedoch ist es interessant das Konzept zu erwähnen.

3.2.4 Regeln basierte Entscheidungsfindung für Interface Adaption

In den Arbeiten “Component-based development of adaptive user interfaces” und “Rule based adaptive user interface for adaptive E-learning system” wird beschrieben, dass der Entscheid für die Umstrukturierung des Interfaces basierend auf Regeln gefällt wird. In der Arbeit “Component-based development of adaptive user interfaces” werden die Regeln dynamisch zur Laufzeit, basierend auf Kontextinformationen generiert. In der Arbeit “Rule based adaptive user interface for adaptive E-learning System” werden die Regeln vorgängig global definiert. Da sich die Regelfindung sich auf Benutzerkontext bezieht, können wir das Konzept nicht vollumfänglich in unser Projekt integrieren. Das allgemeine Prinzip von einem Set an Regeln werden wir jedoch in unserem Projekt verwenden.

4 Adaptive Grid Konzept

4.1 Adaptive Grid Aufbau

4.1.1 Prinzip

Als Einleitung in die Erarbeitung des konkreten Konzeptes, möchten wir die Grundidee erklären. Um das Layout konzeptionell sinnvoll darzustellen, haben wir uns für eine Grid Lösung entschieden. Der Grund für diese Entscheidung ist, dass durch ein Grid die Position eines Elementes am genauesten bestimmt werden kann. Das Ziel ist es, dass wir alle direkten Kinder eines definierten Elementes automatisch anordnen können. Ebenfalls soll diese Anordnung während der Laufzeit durchgeführt werden.

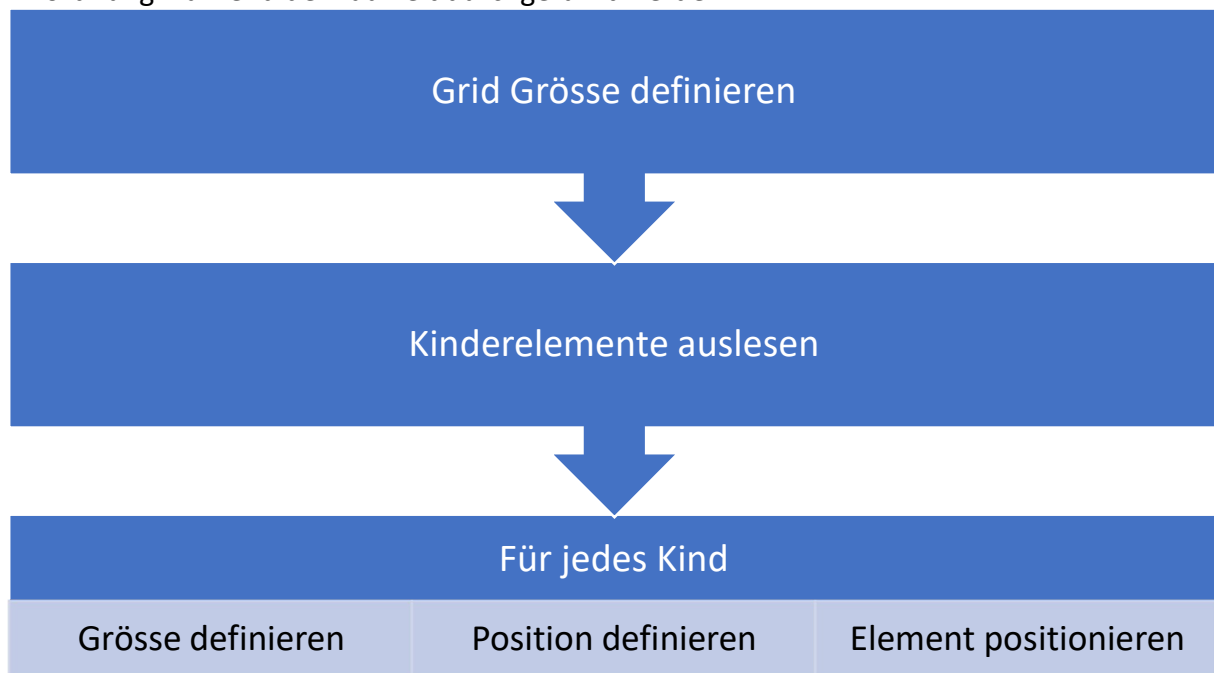


Abbildung 4: Ablauf Anordnung der Elemente

Wie in Abbildung 4 zu erkennen ist, setzt sich der Prozess aus insgesamt fünf Schritten zusammen. Diese bestehen aus zwei allgemeinen Schritten und drei Schritten, die bei jedem Kind individuell durchgeführt werden. Im Folgenden erläutern wir die einzelnen Schritte detailliert: Abbildung 4: Ablauf Anordnung der Elemente

1. Grid Grösse definieren

Als erstes soll definiert werden, welche Grösse das Grid haben wird. Dies bedeutet, wieviele Spalten das Grid aufweisen wird. Entscheidend für diese Definition ist die Fenstergrösse.

2. Alle direkten Kinderelemente auslesen

In diesem Teil des Algorithmus werden alle direkten Kinderelemente ausgelesen. Anschliessend muss das AG wissen, welche direkten Kinder vorhanden sind und welche relevanten Eigenschaften diese Kinder enthalten. Welche Eigenschaften relevant sind, werden wir in folgenden Kapiteln noch genauer definieren.

3. Für jedes direkte Kind werden folgende Aufgaben durchgeführt

a. Die Grösse des Kinderelements wird definiert

Um das Element korrekt platzieren zu können im Grid, muss das Adaptive Grid wissen und/oder entscheiden, welche Grösse das Kinderelement hat. Hier verfolgen

wir ebenfalls verschiedene Ansätze und mögliche Erweiterungen. Im folgenden Kapitel werden wir dies genauer beschreiben

b. Position definieren

Hier wird entschieden, wo im Grid das Element positioniert wird. Hier werden verschiedene Regeln angewendet, um zu entscheiden, an welcher Position im Grid das Element platziert wird.

c. Element positionieren

Als letztes wird das Element im noch mit der definierten Grösse und an der definierten Position eingesetzt.

4.1.2 Aufbau und Variation

Ein Grid kann verschieden aufgebaut sein. Die Aufteilung der Spalten kann z.B. in sechs oder zwölf Spalten erfolgen. Die äussere Breite und die Spaltenabstände des Grids können auch variieren. In diesem Abschnitt legen wir den grundlegenden Aufbau des Grids fest. Ob die genannten Eigenschaften nach Anforderungen geändert werden können und wie flexibel das System ist, werden wir ebenfalls definieren.

Grid Grösse

Das Adaptive Grid basiert auf einem 12 Spalten Grid Raster. Dadurch können die Positionen der einzelnen Elemente auf der X-Achse genauer definiert werden.

Automatische Berechnung der Folgepunkte

Beim Aufbau des Grids werden alle direkten Kinder Elemente des Container Elementes ausgelesen und in das Adaptive Grid platziert. Dabei wird jedem Element eine Zelle zugewiesen. Die Reihenfolge der Elemente entspricht vorerst der Reihenfolge wie sie im DOM sortiert sind.

Elementgrösse

Wir müssen die Grösse jedes Elementes klassifizieren können. Für die Klassifizierung definieren wir in einer ersten Phase, dass die Grösse der Elemente über ein Data-Attribut mitgegeben werden kann. Folgende Grössen sind in einer ersten Umsetzungsiteration vorgesehen:

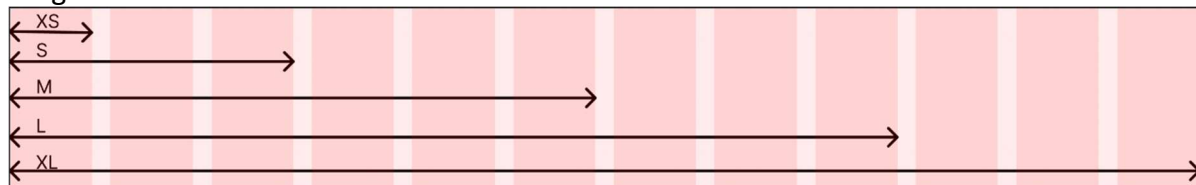


Abbildung 5: Vordefinierte Elementgrößen

Data Attribute werden von allen Browsern unterstützt und erlauben es uns möglichst flexibel und einfach die Grössen der einzelnen Elemente zu definieren. Die Verwendung der Data Attribute ermöglicht uns Plattformunabhängig zu bleiben bzgl. der Grössendefinition. Des Weiteren haben wir in einem Folgeschritt erarbeitet, dass die Grösse kalkuliert wird.

Wie dynamisch bekommen wir die Elementgrösse hin?

Um die Größe der Elemente auf der X-Achse festzulegen, benötigen wir eine vordefinierte Grösse, da die Größe von HTML-Elementen nicht automatisch anhand des Inhalts bestimmt werden kann. In einer zukünftigen Erweiterung des MVP soll dies jedoch möglich sein."

Wie einfach erweiterbar?

Durch diese Umsetzungsart gewährleisten wir, dass unser Adaptive Grid skalierbar ist. Data Attribute können wir einfach mit weiteren Definitionen erweitern. Das Adaptive Grid basiert

auf einem fixen Raster, welche die Grenzen des Layouts spannt. Dabei spielt die Gesamtbreite keine Rolle, da die Spalten sich relativ zur Gesamtbreite anpassen.

Startposition

Nachdem das Adaptive Grid initialisiert wurde und die Kinderelemente in den Zellen des Adaptive Grid platziert wurden, wird für jedes Element eine Startposition berechnet. Die Berechnung der Startposition basiert auf Regeln. Die Regeln werden im entsprechenden Kapitel genauer erklärt (Siehe 4.2).

Erweiterbarkeit der Regeln

Wir werden nicht alle Regeln von Beginn abdecken können. Deshalb werden wir die Regelintegration so vorbereiten, dass eigene Regeln in das Adaptive Grid implementiert werden können.

Dynamische Höhe

Unser Adaptive Grid soll nicht nur in der Breite die Elemente neu anordnen. Auch die Höhe der Elemente soll registriert werden. Ohne die Registrierung der Höhe könnte folgendes Problem auftreten:

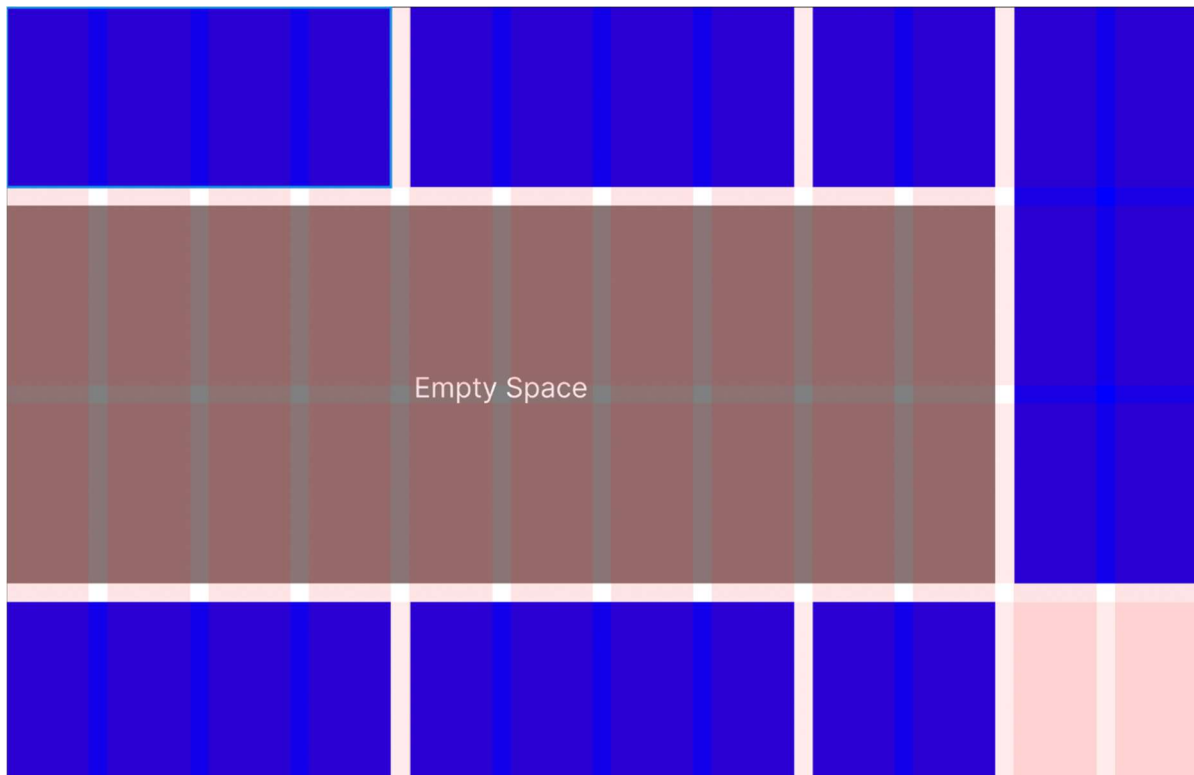


Abbildung 6: Darstellung Empty Space

Wie Sie in Abbildung 6 erkennen können, zeigt der Empty Space die Fläche an, welche wir zu vermeiden versuchen. Um dies zu umgehen, haben wir ein System konzipiert, wie wir dynamisch die Höhe der einzelnen Elemente bestimmen. Um genauer zu sein, bestimmen wir anhand der Höhe der Elemente die Anzahl der Zellen auf der Y-Achse, die ein Element belegen. Wir durchsuchen die Zellen nach dem Element e mit der kleinsten Höhe k_a . Anschliessend prüfen wir das Verhältnis jedes anderen Elementes n zur Höhe k_a . Die Höhe k_a definiert die Höhe jeder einzelnen Zelle. Elemente n , die eine Vielfache y der Höhe k haben, füllen gerade y Zellen auf der Y-Achse. Da wir nicht davon ausgehen können, dass ein fixes Set von Höhen existiert, haben wir einen Puffer p eingeführt. Damit nicht für Elemente n , die nur wenige Pixel grösser als k_a sind, eine zusätzliche Zelle auf der Y-Achse reserviert wird, definieren wir die Puffer Konstante $p = k * (1/2)$. Dieser Puffer dient als

Normalisierung der Höhen. Die Puffer Regel greift nur für Elemente n für die gilt, dass Höhe $n < 2k_a$. Sollte es ein solches Element n geben, wird die neue Zellen Höhe $k_e = k_a + p$ gesetzt.

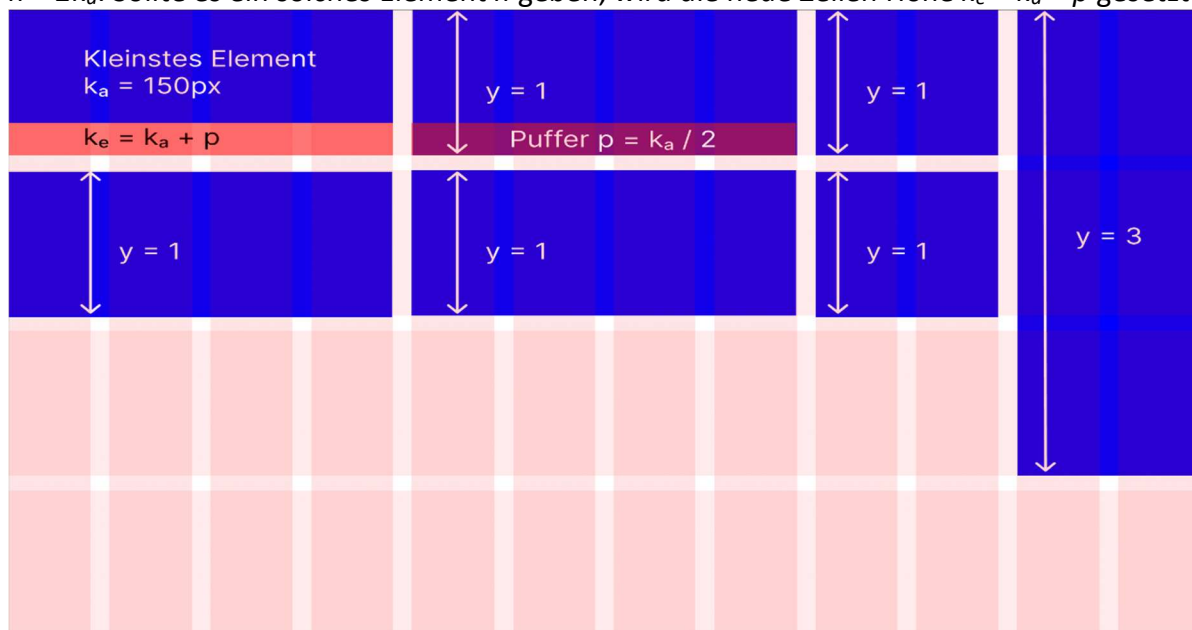


Abbildung 7: Mögliche Anordnung von Elementen

Die blau markierten Zellen sind als besetzt markiert. Das Adaptive Grid weiss dadurch, dass diese Zellen nicht für die Verwendung neuer Elemente verwendet werden können. Bei jeder Änderung der Elemente, welche sich in der Adaptive Grid Zone befinden, muss das Adaptive Grid über die Änderung informiert werden. Dies erreichen wir über das MutationObserver Interface. Über das Interface kann auf Änderungen am DOM (Document Object Model) reagiert werden. Das Adaptive Grid erhält nur Informationen über Änderungen, welche die Child Elemente des Adaptive Grids betreffen.

4.1.3 Intelligente Zellen

Das Adaptive Grid wird in Zellen unterteilt. Die Zellen sollen die Inhaltselemente einkapseln. Durch die Kapselung können den Inhaltselementen Kontextinformationen an das «Adaptive Grid» mitgegeben werden. Für die Speicherung dieser Kontextinformationen wird ein Array verwendet. In den einzelnen Zellen werden die vom Adaptive Grid eingeschlossenen Elemente nicht direkt eingefügt, dafür haben wir sogenannte ProxyCell Objekte erstellt. Ein ProxyCell Objekt kapselt die einzelnen Elemente ein und enthält Metainformationen der Elemente. In dem wir nur Proxy Objekte im Array speichern und nicht die HTML-Elemente selber, können wir gewährleisten, dass wir das originale HTML-Element nicht verändern und bei Bedarf zusätzliche Metainformationen, unabhängig vom HTML-Element, hinzufügen können. Jede ProxyCell Instanz kennt seine Umgebung und hat einen direkten Kommunikationskanal zum Adaptive Grid. Diese Funktionalität wird unter anderem bei der Element-Ordnung benötigt. Folgend finden Sie noch eine Illustration dieses Prinzips.

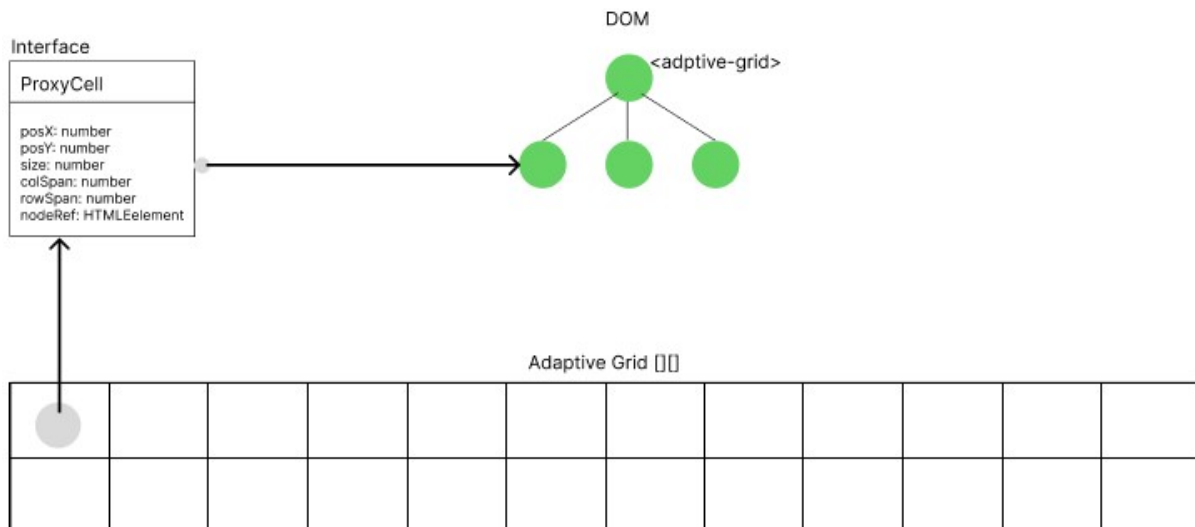


Abbildung 8: Adaptive Grid Model

4.1.4 Ordnung

Jedes Element, das von unserem «Adaptive Grid» verwaltet wird, wird einer bestimmten dynamischen Ordnung zugewiesen. Die Ordnung ist dynamisch, weil sie zur Laufzeit ändern kann. Bei der Generierung des DOMs, werden sämtliche Kind-Elemente des Adaptive Grid der Reihe nach dem Adaptive Grid hinzugefügt. Wie bereits im vorherigen Kapitel erwähnt, werden dabei nicht die HTML-Elemente selbst hinzugefügt, sondern ProxyCell Objekte. Bei der Initialisierung eines ProxyCell Objektes, wird im Konstruktor die Grösse des HTML-Elementes ausgelesen, dass positioniert werden soll. Die Breite wird in das ProxyCell.colSpan Attribut gespeichert. Dieser Prozess wiederholt sich für jedes Kind. Die Ordnung der Elemente verändert sich in dieser Phase nicht.

In einer zweiten Phase werden die Verhältnisse der Elementhöhen zueinander berechnet. Wie dies konkret funktioniert, wird im Kapitel Aufbau beschrieben. Das Verhältnis repräsentiert die Anzahl der Zellen auf der Y-Achse, welches das Element benötigt, um korrekt dargestellt werden zu können. Der Wert wird ebenfalls im ProxyCell Objekt im Attribut ProxyCell.rowSpan gespeichert. Hiermit haben wir alle Informationen gesammelt, die benötigt werden, um mit der Anordnung der Elemente zu starten. Durch unseren Algorithmus werden die Elemente dicht aneinander angeordnet, sodass keine ungewollten Lücken entstehen. Wir nennen den Algorithmus Density-Algorithmus. Der Density-Algorithmus holt sich aus einem Set von Elementen in jeder Iteration die nächstgrössten Elemente und platziert sie an einer bestimmten Position im Grid. Die Grösse eines Elementes wird anhand der korrespondierenden ProxyCell Attribute rowSpan und colSpan gemessen.

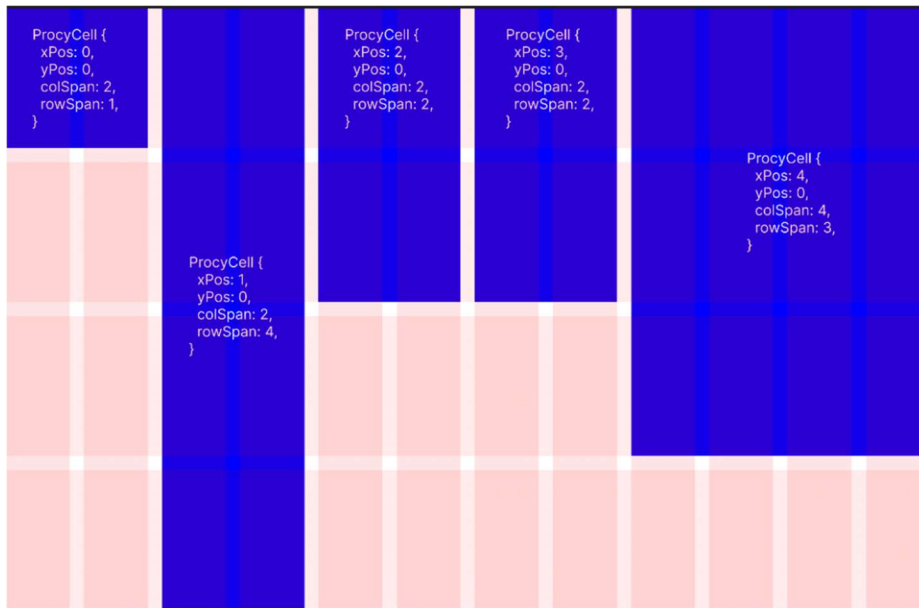


Abbildung 9: ProxCells mit elementspezifischen Werten

In Abbildung 9 ist ersichtlich, wie die ProxyCells mit elementspezifischen Werten befüllt wurden. Der Density-Algorithmus würde dieses ProxyCell Array folgendermassen anordnen:

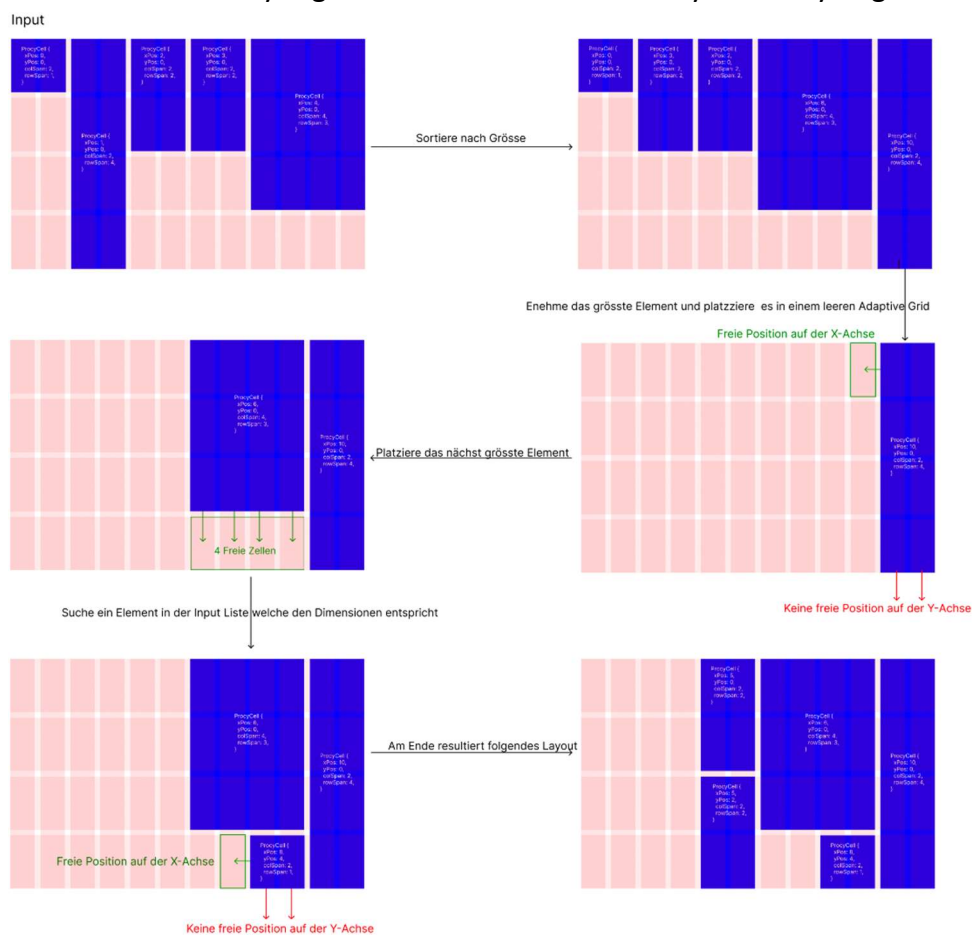


Abbildung 10: Ordnungsprozess

Der Density Algorithmus stellt den Kern unseres Ordnungssystems dar. Dieser Kern wird erweitert durch Regeln, welche es ermöglichen, eine benutzerdefinierte Anordnung zu definieren. Des Weiteren können mit den Regeln Ausnahmefälle abgedeckt werden.

4.2 Regeln

Regeln sind Variablen, welche die Ordnung der Elemente innerhalb des «Adaptive Grid» definieren. Über Regeln kann der Density Algorithmus, welcher die Basis unseres Ordnungssystems bildet, mit zusätzlicher Logik erweitert werden. Uns ist bewusst, dass wir niemals alle möglichen Layouts, die sich durch das Kombinieren von verschiedenen Elementen ergeben können, abdecken. Deshalb ist es angedacht, das Set an Regeln erweitern zu können (Siehe Kapitel 4.3). Um eine optimale Skalierbarkeit zu gewährleisten, entwickeln wir ein Schema für die Definition von Regeln. Somit kann der Adaptive Grid einfach mit Regeln erweitert werden, solange die Regeln das definierte Schema anwenden.

4.3 Skalierbarkeit

Beim Konzept von Adaptive Grid besteht die Herausforderung, dass eine grosse Anzahl an möglichen Elementkombinationen vorkommen kann. Die Dokumentation und Abdeckung all dieser möglichen Kombinationen ist unmöglich, da unendlich viele Elemente vorkommen können im Adaptive Grid. Um dieses Problem zu umgehen, möchten wir die Möglichkeit erschaffen, die bestehenden Regeln für das Design mit eigenen zu ergänzen. Um dies zu lösen, kann einerseits der Code mit weiteren Funktionen ergänzt werden. Andererseits besteht die Möglichkeit, die Menge an Regeln zu vergrössern durch Konfigurationsdateien im JSON-Format. Durch dies werden weitere Möglichkeiten erschaffen, erhaltene Elementkombinationen abzudecken. Zusätzlich ermöglicht die Kapselung der Elemente in ProxyCells eine erleichterte Erweiterung der Funktionalität des Adaptive Grids, ohne die verknüpften HTML-Elemente verändern zu müssen.

4.4 Umgang mit Fehler

Da das «Adaptive Grid» als Steuerelement für den gesamten Inhalt einer Seite fungieren wird, müssen wir das Verhalten bei bestimmten Ausnahmesituationen, welche unbehandelt zu Exceptions führen, definieren.

1. Eine grosse Anzahl an direkten Kinderelementen

Bei einer grossen Anzahl an direkten Kinderelementen können verschiedene Probleme entstehen, wie grosse Ressourcenlast, Performance-Issues, etc. Jedoch kann ebenfalls nicht ein einfaches Lazy-Loading Feature implementiert werden. Denn beim Lazy-Loading wird eine bestimmte Anzahl der DOM-Elemente geladen und der Rest wird nicht angezeigt, bis man den nächsten Teil nachladen möchte. Jedoch bearbeitet unser Schema nicht die Reihenfolge der Elemente im DOM-Baum, sondern passt einfach dessen «Style» an, um das Element an einem bestimmten Ort zu platzieren. Adaptive Grid entfernt keine Objekte und fügt ebenfalls keine hinzu. Deshalb haben wir eine Limite an Kinderelementen definiert, welche variabel auf jedes System angepasst werden kann. Wenn diese Limite erreicht wird, wird einerseits eine Meldung dargestellt, welche den Benutzer informiert, dass es noch mehr Elemente gibt und diese jedoch nicht dargestellt werden aus Performance-Gründen. Hier muss jedoch angefügt werden, dass wenn Adaptive Grid dies bemerkt, sollte dies mehr eine Bemerkung für den Entwickler sein, und eine Lösung sollte implementiert werden, sodass die Limite an Elementen nicht erreicht wird.

2. Adaptive Grid kann aus bestimmten Gründen nicht ausgeführt werden

Es kann aus verschiedenen Gründen das Problem entstehen, weswegen das Adaptive Grid eine Exception wirft oder generell nicht geladen werden kann. In diesen Fällen werden die

Elemente einfach ohne die Bearbeitung von Adaptive Grid dargestellt. Falls eine Exception geworfen wird, wird diese geloggt und kann eingesehen werden.

3. Keine Kinderelemente sind enthalten

Wenn das Adaptive Grid keine Kinderelemente hat, wird der reservierte Platz für das Adaptive Grid immer noch dargestellt. Es kann eine Meldung dargestellt werden, falls dies gewünscht ist.

4.5 Performance

Für unsere Adaptive Grid Lösung ist eine gute Performance essenziell. Deshalb versuchen wir unseren Density Algorithmus mit einer möglichst geringen Komplexitätsklasse umzusetzen. Mit dieser Komplexitätsklasse sollte das dynamische neu Anordnen zur Laufzeit keine bemerkbaren Performance Einbrüche ergeben.

Einen anderen kritischen Performance Aspekt, könnten unsere ProxyCell Objekte darstellen. Die Generierung von Objekten ist in Bezug auf die Performance ein teurer Befehl. Bei vielen Kind-Elementen eines Adaptive Grid Containers könnte dies das System beeinträchtigen.

Nicht nur das Erzeugen der Objekte, sondern auch die bidirektionalen Kommunikationskanäle, die zwischen dem Adaptive Grid und den einzelnen Zellen aufgebaut werden, stellen dabei ein Risiko dar. Da wir aber von einem UI basierten Use Case ausgehen, sollten die generierten ProxyCell Objekte eine überschaubare Menge sein. Aus diesem Grund können wir auch dieses Risiko ausklammern. Trotzdem werden wir die Objekte so schlank wie möglich halten und die Kommunikation zwischen den Zellen und dem Adaptive Grid minimalisieren.

5 Implementierung

5.1 Ausgangslage

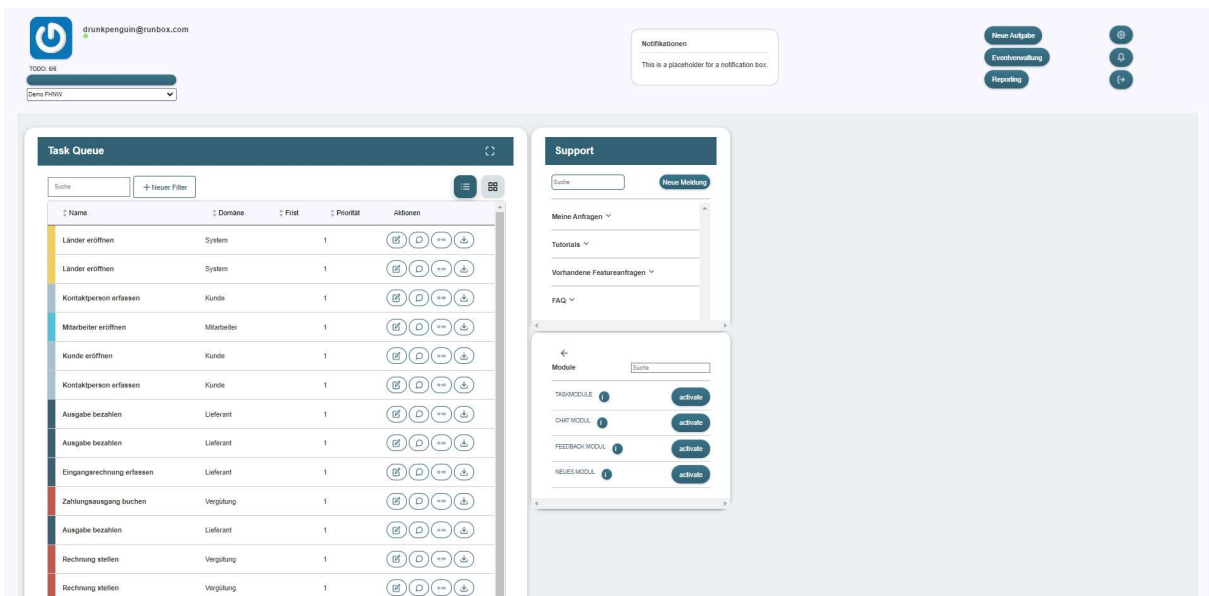


Abbildung 11: Dashboard Enablerr

Enablerr wird von den Herstellern als die revolutionäre Business Solution bezeichnet [1]. Die Softwarearchitektur ist radikal anders aufgebaut. "Enablerr unterscheidet sich deutlich von allem, was du heute kennst und bietet noch viel mehr: Erweiterbarkeit ohne Grenzen, Wartungsfrei und ohne lästige Update- oder Upgrade-Unterbrüche." [1] Enablerr befindet sich zurzeit noch in Entwicklung und ein grosser Teil des UI wird automatisch generiert,

abhängig von verschiedenen Parametern. Dies erschwert die zuverlässige Generation eines User-Interface, welches eine angenehme User-Experience ermöglicht. Wir möchten als POC des Adaptive Grid das Dashboard von Enablerr mit Adaptive Grid implementieren und dessen Implementation evaluieren. Parallel zu der Integration des Dashboards untersuchen wir Enablerr auf weitere mögliche Einsatzmöglichkeiten des Adaptive Grid. Spezifisch gehen wir hier noch auf die Generation von Tasks ein.

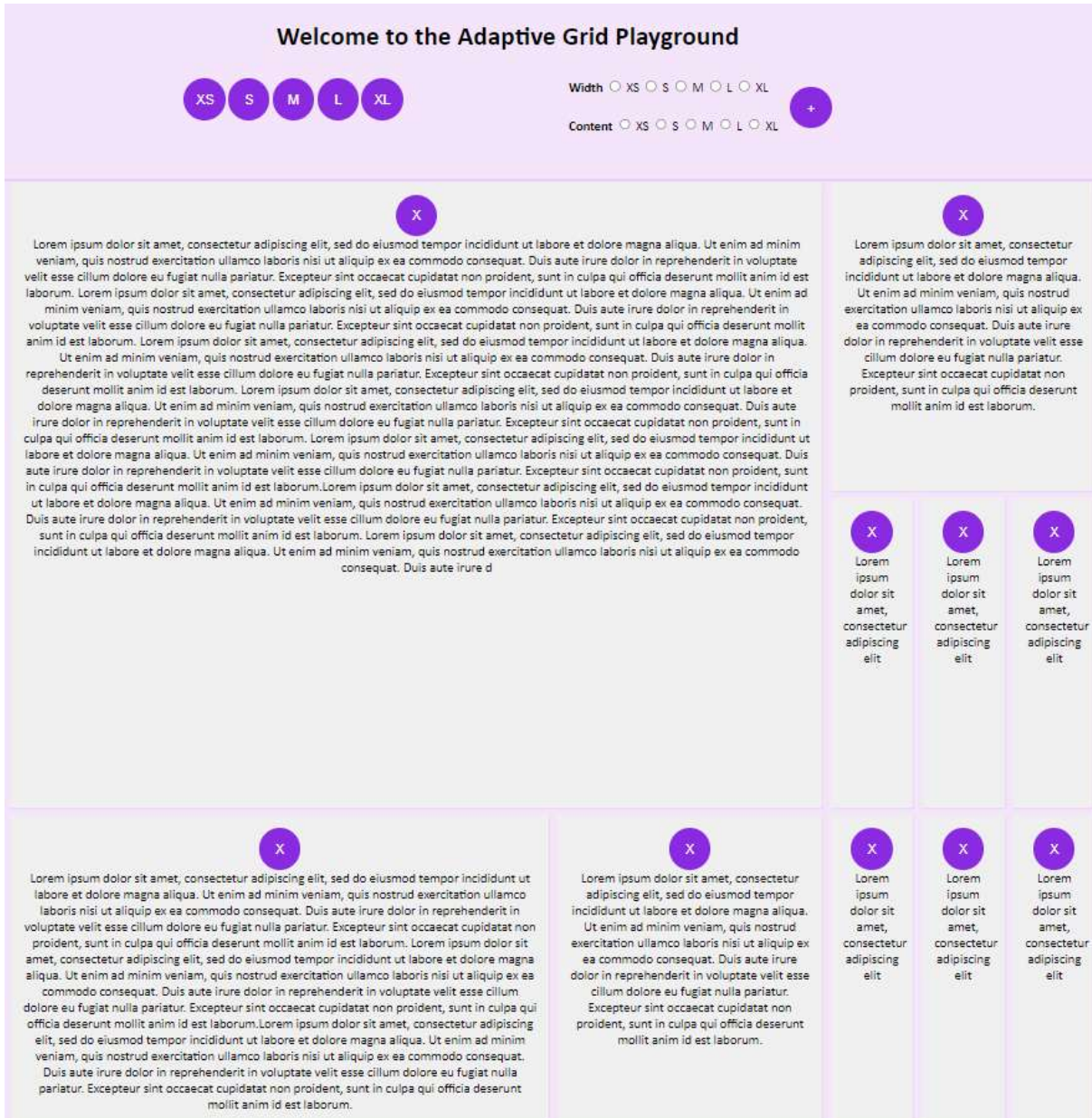
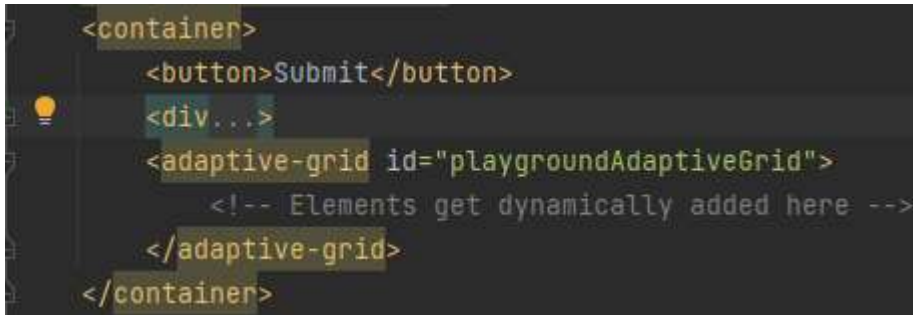


Abbildung 12: Adaptive Grid Playground

Des Weiteren entwickeln wir den «Adaptive Grid Playground». Dies ist eine Webapplikation, durch welche wir verschiedene Szenarien testen können und ebenfalls Shareholders die verschiedenen Möglichkeiten und Features des Adaptive Grid aufzeigen. Das Adaptive Grid Playground sollte darüber hinaus als Testumgebung für alle Entwickler dienen, die das Adaptive Grid verwenden. Es ermöglicht ihnen, verschiedene Layouts auszuprobieren und zu optimieren.

5.2 Technische Herausforderung

5.2.1 Universalität



```
<container>
  <button>Submit</button>
  <div...>
    <adaptive-grid id="playgroundAdaptiveGrid">
      <!-- Elements get dynamically added here -->
    </adaptive-grid>
  </div>
</container>
```

Abbildung 13: Integration des Adaptive Grids in eine HTML Seite

Unser Ziel ist es, dass das Adaptive Grid in allen JavaScript Frameworks und/oder Vanilla Javascript Applikationen eingesetzt werden kann. Dies bedeutet, dass wir uns nicht auf Framework spezifische Methoden und Funktionen verlassen dürfen. Aus diesem Grund haben wir uns dazu entschieden, ein Benutzerdefiniertes HTML-Element zu erstellen, welches als Webkomponente fungiert. Eines der Hauptmerkmale der Webkomponenten ist die Möglichkeit, benutzerdefinierte Elemente zu erstellen, die Ihre Funktionalität auf einer HTML-Seite kapseln, anstatt sich mit einem langen, verschachtelten Stapel von Elementen begnügen zu müssen, die zusammen eine benutzerdefinierte Seitenfunktion bereitstellen. Eine Beispielintegration ist in der Abbildung 13 ersichtlich.

5.2.2 Observers

Eine Kernfunktion unserer Adaptive Grid Implementation ist die Reaktivität des Inhalts zur Laufzeit. Das Adaptive Grid reagiert auf von uns bestimmte Veränderungen im DOM und regeneriert das Layout, passend für die neuen Parameter. Damit wir auf Veränderungen im DOM reagieren können, verwenden wir die Mutation Observer Schnittstelle [31]. Aus Performance Gründen mussten wir zwei verschiedene Mutation Observer implementieren. Der Erste reagiert lediglich auf das Hinzufügen und Entfernen von Elementen innerhalb des Adaptive Grids. Unser Adaptive Grid soll aber nicht nur auf das Hinzufügen oder Entfernen von Elementen reagieren, sondern auch auf sämtliche Veränderungen von bereits platzierten Elementen. Sollte zum Beispiel die Höhe eines Elementes ändern, wird ein zweiter Observer aktiviert, der State-Observer, welcher das Adaptive Grid darüber informiert, dass sich ein Element verändert hat. Unabhängig vom Observer landen Elemente in einer Node-Pipeline, welche wir implementiert haben. Diese dient als Queue für Elemente, welche vom Adaptive Grid verarbeitet werden sollen. Neue Elemente werden aus der Node-Pipeline direkt in den Adaptive Grid Zellen Speicher geschrieben. Elemente, welche vom State Observer hinzugefügt werden, werden nur hinzugefügt, wenn sie im Zellen Speicher noch nicht existieren. Sollten Sie bereits existieren, werden die alten mit den neuen Proxy Zellen ersetzt. Wie bereits erwähnt, mussten wir den State-Observer vom Hauptobserver aus Performance Gründen trennen. Damit unser Adaptive Grid auch wirklich reaktiv wirkt, müssen wir nicht nur auf State Änderungen der direkten Kinder reagieren, sondern müssen den gesamten Knotenbaum des Adaptive Grids observieren. Das bedeutet, dass alle State Änderungen aller Kinder des Adaptive Grids berücksichtigt werden müssen. In der Abbildung 14 ist ersichtlich was damit gemeint ist.

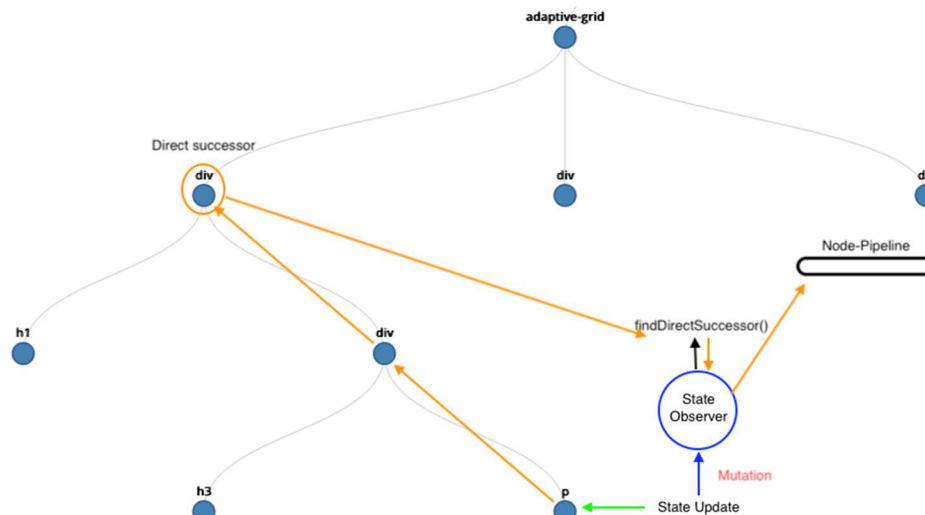


Abbildung 14: Der State des `<p>` Tags verändert sich. Dadurch wird der State Observer aktiviert. Da wir nur direkte Nachkommen des Adaptive Grid Elementes verarbeiten, muss im DOM der Baum so weit nach oben geklettert werden, bis ein Knoten gefunden wird, welcher der direkte Nachkomme des Adaptive Grids ist. Dafür haben wir die Methode `findDirectSuccessor()` erstellt. Das gefundene Element wird anschliessend vom Observer in die Node-Pipeline hinzugefügt.

Da sich der Status der Kinder auch beim Hinzufügen neuer Elemente ändert, haben wir eine Steuerung eingebaut, die beim Hinzufügen oder Entfernen von Elementen den State-Observer deaktiviert. Nach Abschluss der Arbeit, wird er wieder reaktiviert.

5.2.3 Node-Pipelines

Eine Mutation ist ein Objekt, dass vom Observable bei einer Änderung im Adaptive Grid an die Subscriber gesendet werden. Bei mehreren Änderungen innerhalb des Adaptive Grids werden mehrere solcher Mutationen gesendet. Damit wir nicht bei jeder Mutation das Adaptive Grid neu generieren müssen, haben wir zwei Node-Pipelines eingeführt. Eine Node-Pipeline für neue Elemente und eine für Elemente die gelöscht werden. Bei Änderungen im Adaptive Grid werden in einem ersten Schritt alle Mutationen geprüft und gefiltert. Adaptive Grid relevante Mutationen landen in den entsprechenden Node-Pipelines. Sobald alle Mutationen einer Nachricht geprüft wurden und der Node-Pipeline hinzugefügt wurden, rekonstruieren wir das Layout basierend auf den neuen Nodes.

5.2.4 Zellspeicher und ProxyCell Objekte

Im Zellspeicher werden ProxyCell Objekte gespeichert. Gefilterte Mutationen, welche sich in einer Node-Pipeline befinden, werden aus der Node-Pipeline entnommen und in ein ProxyCell Objekt gekapselt. Dabei werden für das Adaptive Grid relevante Informationen aus der Mutation extrahiert und im ProxyCell Objekt abgelegt. Jedes HTML-Element einer Mutation wird von genau einer ProxyCell gekapselt. Die ProxyCell wird anschliessend im Cell-Storage abgelegt. Das Cell-Storage ist ein eindimensionales Array, welches immer den aktuellen Stand der Daten enthält, bevor die ProxyCell Objekte im Adaptive Grid platziert werden. Im Cell-Storage können Daten nicht redundant abgelegt werden. Der Identifier jeder Proxyzelle ist dabei der DOM-Knoten selbst. Indem wir die Node-Pipelines in Anfragen zum Hinzufügen/Aktualisieren und Löschen trennen, können wir die Aufwände beim Löschen reduzieren.

5.2.5 Navigation innerhalb des Adaptive Grids

Die Navigation innerhalb des Adaptive Grids ist eine Kernfunktion unseres Adaptive Grids. Das Adaptive Grid muss übergeordnet immer den Zustand jeder einzelnen Zelle kennen. Wir unterscheiden aktuell drei Zustände:

- Frei

- Belegt
- Reserviert

Damit die Navigierung überhaupt stattfinden kann, müssen die Adaptive Grid Grenzen gesetzt sein. Auf der X-Achse gehen wir aktuell immer von einer fixen Spaltenanzahl von 12 Spalten aus. Diese Zahl ist jedoch variabel gesetzt, um Weiterentwicklungen zu vereinfachen. Auf der Y-Achse wird die Zeilen-Anzahl dynamisch anhand des höchsten Elementes im Cell-Storage definiert. Sind diese Grenzen gesetzt, wird ein zweidimensionales Array aufgebaut. Die Länge des äusseren Arrays entspricht der Höhe des höchsten Elementes geteilt durch die Zeilen Höhe, welche in der init-Methode definiert werden kann. Für jede Array Position wird ein weiteres Array mit der Länge 12 eingefügt. Die inneren Arrays bilden die X-Achse ab. Für die Navigation innerhalb des Adaptive Grids, haben wir den Density Algorithmus programmiert. Das Konzept dahinter kann im Kapitel 4.1.4 beschrieben. Der Algorithmus wird allerdings nur in der Ordnungsphase verwendet. Über Regeln können Plätze innerhalb des Adaptive Grids priorisiert werden. Dadurch kann der Entwickler bestimmte Bereiche im Adaptive Grid für bestimmte Elemente reservieren. Für Reservationen haben wir ein spezielles Register erstellt, welches uns erlaubt mit einer Komplexitätsklasse von $O(1)$ die reservierten Elemente im Adaptive Grid zu platzieren.

Bei der Kapselung der Elemente in Proxymzellen, wird geprüft, ob einem Element eine Reservationsregel enthält. Ist dies der Fall, werden die notwendigen Informationen für die betroffene Proxymzelle direkt im Reservationsregister abgelegt. Bei der Generierung des Layouts, wird zuerst das Reservationsregister abgearbeitet, indem die entsprechenden Elemente direkt im Raster an der entsprechenden Stelle platziert werden.

Im Kapitel gehen 5.7.4 gehen wir genauer auf die Unterschiede der beiden Varianten ein.

5.2.6 CSS-Generierung

Es ist uns wichtig, die Integration des Adaptive Grids so einfach wie möglich zu gestalten. Deswegen muss das Adaptive Grid CSS nicht über eine externe CSS-Datei eingebunden werden, denn es wird im Build-Prozess per JavaScript generiert. Während der Implementierung des CSS, stiessen wir auf das Problem, dass unser CSS, während dem Build-Prozess die Breite und Höhe der Elemente verfälschte. Dadurch entstand nach der Generierung eine fehlerhafte Darstellung der einzelnen Elemente. Um dies zu umgehen, hätten wir ein kleines Adaptive Grid CSS Framework erstellen können, womit der Entwickler die Dimensionen der Elemente noch vor dem Generieren des Layouts spezifizieren muss. Dies würde aber einen Mehraufwand für den Entwickler bedeuten, denn er müsste für jedes Element im Vorfeld die Klassen analysieren und entsprechend setzen. Unser Ziel ist, den Aufwand für die Integration des Adaptive Grids so gering zu halten wie nur möglich. Aus diesem Grund haben wir ein System implementiert, welches vor dem Auslesen der Element-Dimensionen, das CSS unseres Adaptive Grids deaktiviert. Nachdem sämtliche Metadaten gesammelt wurden und der eigentliche Build-Prozess startet, wird das CSS reaktiviert. Mit diesem System können wir gewährleisten, dass die HTML-Elemente ohne zusätzlichen Aufwand für den Entwickler, korrekt vom Adaptive Grid interpretiert und dementsprechend auch korrekt angezeigt werden. Dadurch werden die Elemente automatisch in ihrer ursprünglichen Größe verarbeitet, ohne dass der Entwickler zusätzlichen Aufwand betreiben muss.

5.2.7 Berechnung der Zellen und der rowSpan Einträge

Im Konzept haben wir beschrieben, dass wir für die Berechnung der Zeilenhöhe das kleinste Element im Cell-Storage suchen. Die Höhe des kleinsten Elementes sollte als Zeilenhöhe verwendet werden. Während der Implementierung haben wir bemerkt, dass dieser Ansatz nicht optimal ist. In der Abbildung 15 ist dieses Problem beschrieben. Das kleinste Element ist in der Abbildung blau markiert. Dieses Element definiert die Höhe jeder Zelle. Der rot markierte Bereich in der Abbildung visualisiert das eigentliche Problem, dass dadurch entsteht. Das Adaptive Grid definierte die Dimensionen der Elemente, in dem das Element über die Zellen gezogen wird. In diesem Beispiel ist ersichtlich, dass sich das Chat Modul nur ein paar Pixel über die dritte Zeile hinausstreckt. Das Adaptive Grid streckt das Element trotzdem über die gesamt dritte Zeile hinaus. Je höher das kleinste Element, umso wahrscheinlicher, dass die Höhe eines Elementes sichtbar verfälscht wird.

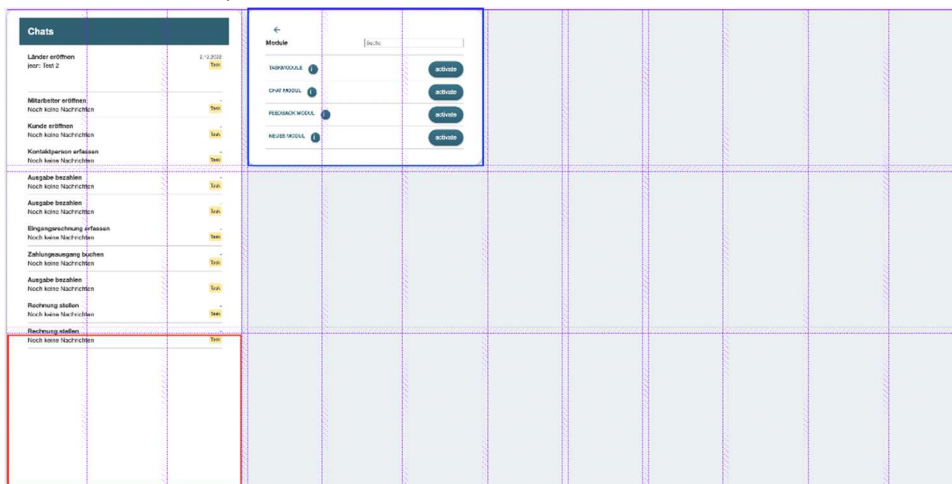


Abbildung 15: Problemdarstellung des kleinsten Elementes und der Zeilenhöhe

Aus diesem Grund definieren wir die Zeilenhöhe nicht anhand der Höhe des kleinsten Elementes, sondern setzen Sie von Anfang an auf einen möglichst kleinen Wert.

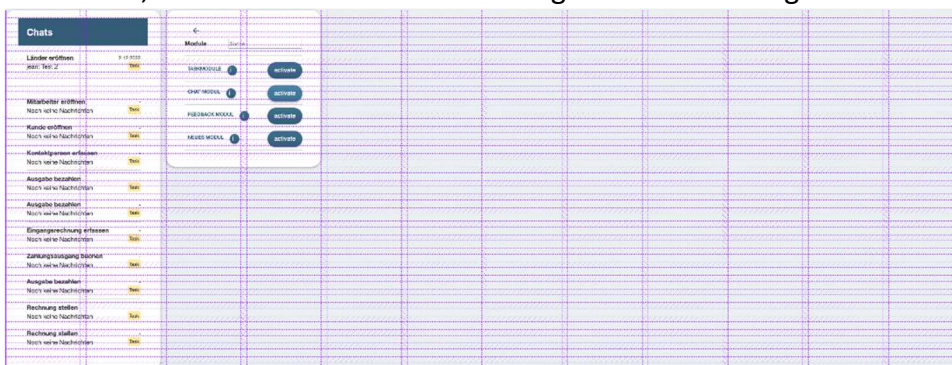


Abbildung 16: Visualisierung des Grids wenn die Zeilenhöhe fix gesetzt wird

In der Abbildung 16 ist die Zeilenhöhe auf 20px gesetzt. Dadurch wird folgendes Grid Layout erzeugt. Beim Chatmodul ist ersichtlich, dass die Höhe des Chatmoduls nicht mehr sichtbar verfälscht wird. Dies bringt allerdings einen Nachteil in Bezug auf die Performance mit sich. Je kleiner die Zeilenhöhe, umso mehr Zeilen werden im Adaptive Grid erzeugt, was die Anzahl der Aufrufe im Aufbauprozess des Adaptive Grids erhöht. Damit der Entwickler selbst abschätzen kann, was für seine Applikation am besten ist, haben wir eine Konfigurationsmöglichkeit implementiert, durch die der Entwickler die Zeilenhöhe bei der Initialisierung selber definieren kann. Wenn der Entwickler weiss, dass die Interaktivität des

Bereiches eher gering ist, könnte er die Zeilenhöhe auf 1px setzen, wodurch die Höhe der Elemente praktisch nicht verfälscht wird.

5.2.8 Abstand System

Die Integration eines einheitlichen Abstand Systems ist für ein visuell ansprechendes Layout von zentraler Bedeutung. [32] Das CSS Grid Layout bietet die Möglichkeit, die Abstände zwischen den Spalten und Zeilen per CSS zu definieren. Die CSS-Definitionen werden col-gap und row-gap genannt. Unser Adaptive Grid greift auf diese Definitionen zurück. Bei der Implementierung stießen wir jedoch auf das Problem, dass der row-gap für jede Zeile der Höhe eines Elementes hinzugefügt wurde. Der col-gap wurde der Gesamtbreite des Adaptiv Grids hinzugefügt. Abbildung 17 stellt dieses Problem dar.

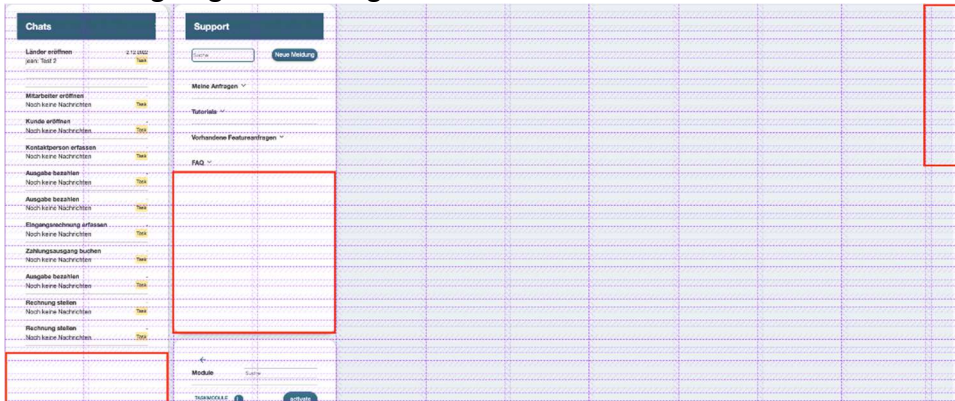


Abbildung 17: Fehlerhafter Generierung der Containerbreite und der Element Höhen durch den row und col-gap.

Damit die Gesamtbreite korrekt gesetzt wird, haben wir im CSS die native calc() Funktion verwendet. Bei der Spalten Definition ziehen wir per calc() Funktion den col-gap ab.

```
grid-template-columns: repeat(${AdaptiveGrid.gridSize}, calc(8.333% - ${this.colGap}px));
```

Abbildung 18: Korrektur der Gesamtbreite mit Hilfe der calc() Funktion

Damit der Element-Höhe der row-gap nicht addiert wird, addieren wir bei der Berechnung der Zeilenverteilung eines Elementes dem Subtrahenden den row-gap.

```
setCellRowSpan() {
  for (let i = 0; i < this.cellStorage.length; i++) {
    const proxyCell = this.cellStorage[i];
    if (proxyCell === undefined) break;
    if (Object.hasOwn(proxyCell, 'size')) {
      const rowSpan = Math.ceil(proxyCell.size / (this.cellHeight + this.rowGap));
      proxyCell.rowSpan = rowSpan;
    }
  }
}
```

Abbildung 19: Korrekte Ausgabe der Höhe aller Elemente

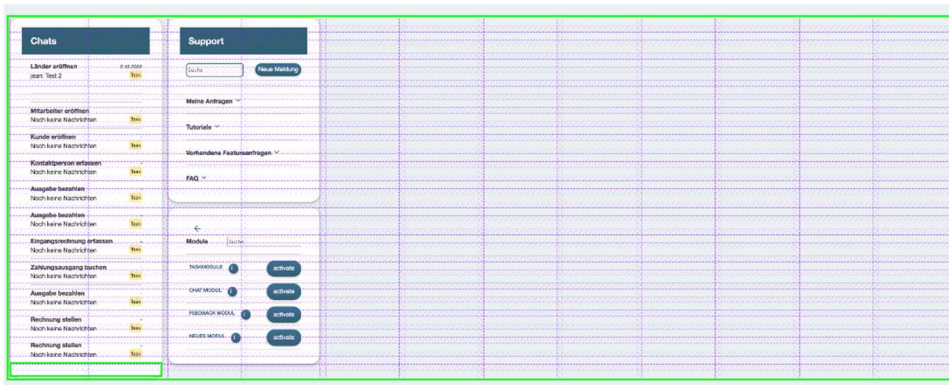


Abbildung 20: Nach den Korrekturen aus der Abbildung 18 und Abbildung 19, wird das Grid Layout korrekt generiert und der Zeilen und Spaltenabstand ist korrekt definiert.

Der Abstand kann als Parameter vom Entwickler mitgegeben werden. Das Adaptive Grid adaptiert das Layout dem definierten Abstand entsprechend.

```
onMount(async () => {
  document.getElementById('adGridDashboard').init(20, 50, 50);
});
```

Abbildung 21: Initialisierung des Adaptive Grid mit col-span und row-span

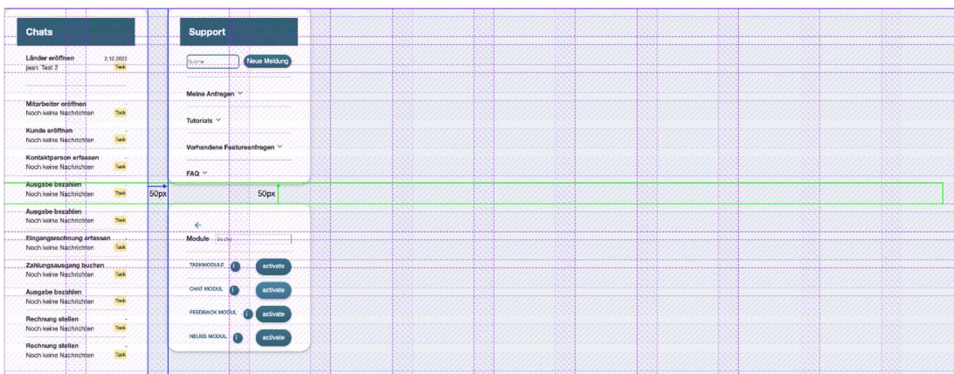


Abbildung 22: Visualisierung der Abstände nach der Layout Generierung

5.2.9 Konflikte mit externem CSS

Unser Adaptive Grid arbeitet nicht mit einem Shadow-DOM. Dadurch gelten globale CSS-Definitionen einer Seite auch für Elemente, die vom Adaptive Grid verwaltet werden. Dies bringt jedoch den Nachteil mit sich, dass es Überschneidungen im CSS geben kann. Gerade wenn die Applikation mit *!important* Statements arbeitet, kann sich ereignen, dass Adaptive Grid interne CSS-Definitionen von aussen überschrieben werden und dadurch das Layout fehlerhaft dargestellt wird. Eine Lösung, die in allen Fällen wirkt, konnten wir aus zeitlichen Gründen nicht finden. Für die Enablerr Plattform haben wir betroffene CSS-Definitionen ebenfalls mit *!important* kennzeichnen müssen.

5.3 Methodik

Da wir bei der Implementation zwei Projekte parallel implementierten (Playground und Enablerr), mussten wir die Synchronisation des Algorithmus sicherstellen zwischen beiden Projekten. Deshalb war Kommunikation sehr wichtig und ebenfalls Koordination. Denn die Gefahr bestand, dass für Enablerr der Algorithmus erweitert wird, diese Erweiterung jedoch nicht eingebunden wird in den Playground. Wir arbeiteten während der

Implementierungsphase mit der Technik des Agile Development. Wir führten daily-Meetings durch, in welchen wir immer evaluierten, ob unser aktueller Plan immer noch zielführend ist, welche Features wir als nächstes implementieren möchten und welche Probleme/Bugs unsere Aufmerksamkeit benötigen. Unsere groben Meilensteine waren wie folgt:

1. Aufbau der Webapplikation "Adaptive Grid Playground"
2. Erste Version des Density-Algorithmus und Adaptive Grids entwickeln
3. Algorithmus Integration übernehmen in die Enablerr Plattform
4. Adaptive Grid erweitern mit dem Enablerr Use Case
 - a. Parallel hierzu Bugfixing des Density Algorithmus im Playground
5. Code-Refactoring durchführen

5.4 Technologien und Frameworks

Wie wir im Kapitel 5.2.1 schon erwähnen, war unser Ziel die universale Einsatzbarkeit des Adaptive Grids in der Welt von JavaScript. Deshalb arbeiteten wir bzgl. der Adaptive Grids Implementierung ausschliesslich mit **HTML**, **CSS** und **JavaScript**. Für die Speicherung und Code-Koordination verwendeten wir **GitHub** [33]. Das Frontend der Enablerr Plattform wurde mit **Svelte** [34] entwickelt.

5.5 Integration

Unsere Implementierung des Adaptive Grids ermöglicht eine sehr einfache Integration in jede Webseite, unabhängig davon, welches JavaScript Framework für die Generierung des Frontend verwendet wird. Bei klassischen HTML-Seiten muss lediglich eine JS-Datei im Header oder Footer der Webseite per Skript Tag hinzugefügt werden. Wichtig ist dabei, dass vor der Initialisierung des Adaptive Grids der DOM bereits geladen wurde, deshalb empfehlen wir, die JS-Datei asynchron oder im Footer der Seite zu laden.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <script type="module" src="adaptive-grid.js"></script>
</head>
```

Abbildung 23: Integration der Webkomponente per Skript Tag

Das Adaptive Grid wird anschliessend vom Browser als natives HTML-Element erkannt und kann dementsprechend auch im Code platziert werden.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <script type="module" src="adaptive-grid.js"></script>
</head>
<body>
  <adaptive-grid>
    <!-- Elemente werden dynamisch im Raster platziert -->
  </adaptive-grid>
</body>
</html>
```

Abbildung 24: Platzierung des <adaptive-grid> HTML-Elementes

Das Adaptive Grid kann ebenfalls in sämtlichen JS-Frameworks wie Svelte, React oder angular implementiert werden. Für uns war in erster Linie die Integration in das Frontend der Enablerr Plattform relevant. Daher zeigen wir die Integration in das Svelte Framework auf.

```
import './adaptiveGrid/adaptive-grid'
import {onMount} from 'svelte';

onMount(async () => {
  document.getElementById('adGridDashboard').init(20, 15, 15);
});
```

Abbildung 25: Einbindung Svelte

Wie bereits erwähnt ist es wichtig, dass das Adaptive Grid nach dem vollständigen Laden des DOMs initialisiert wird. Svelte bietet dafür die onMount Lifecycle Hook an. Damit wir das Adaptive Grid verwenden können, müssen wir die JS-Datei und die onMount Hook per import Statement importieren. Anschliessend kann dem onMount Hook eine asynchrone Callback Funktion mitgegeben werden, welche das Adaptive Grid initialisiert. Der init-Methode können Konfigurationsparameter mitgegeben werden, welche dem Entwickler eine gewisse Flexibilität anbietet. Folgende Konfigurationen können mitgegeben werden:

- Zeilen Höhe jeder Zeile des Adaptive Grids
- Abstand zwischen den einzelnen Spalten in Pixel
- Abstand zwischen den einzelnen Zeilen in Pixel

Im HTML muss lediglich noch das <adaptive-grid> HTML-Element platziert werden. Damit wir einen Vergleich zur vorherigen Integration per Svelte Grid haben, zeigen die folgenden Abbildungen die Integration per Svelte Grid und die Abbildung per Adaptive Grid auf.

```
<section id="main">
  <Grid fillSpace={false} bind:items={module} rowHeight={120} let:dataItem {cols} >
    {#if dataItem.id !== 9999}
      <Button on:click={() => remove(dataItem)} style="height:45px; margin-top: 0.7em; color: white"
        class="close">
        <XIcon></XIcon>
      </Button>
    {/if}
    <Module {dataItem} {settings} functionProp={(item) => addModule(item)}></Module>
  </Grid>
</section>
```

Abbildung 26: Code der Dashboard Komponente vor Integration

```
<section>
  <adaptive-grid id="adGridDashboard">
    {#each module as dataItem}
      {#if dataItem !== undefined && dataItem !== null}
        <Module removeModuleFn={remove} {dataItem} {settings} functionProp={(item) => addModule(item)}></Module>
      {/if}
    {/each}
  </adaptive-grid>
</section>
```

Abbildung 27: Code der Dashboard Komponente nach Integration

5.5.1 Svelte Grid ersetzen

Als erstes mussten wir das vorhandenen Svelte-Grid [35] durch unser Adaptive Grid ersetzen. Die Herausforderung in diesem Punkt war die Filterung relevanter Attribute. Wir

mussten die applikationsrelevanten Attribute trennen von den Funktionen und Attributen, welche für die Konfiguration des Svelte Grid existierten. Ebenfalls mussten wir verschiedene Funktionen anpassen, welche während der Laufzeit das Svelte Grids verändern. Nach dieser Filterung konnten wir in der Komponente des Svelte Grid ersetzen durch den HTML-Eintrag des Adaptive Grids.

5.5.2 Ungewollte Kinderelemente

Das Prinzip des Adaptive Grids besteht darin, dass alle direkten Kinderelemente einer Position innerhalb des dynamisch generierten Grids zugeordnet werden. Bei Enablerr mussten wir jedoch feststellen, dass verschiedene direkte Kinderelemente vorhanden sind, welche nicht beachtet werden sollen. Einerseits wurde für jedes Modul im Dashboard einen «Close-Button» kreiert, welcher jedoch im DOM auf derselben Höhe ist wie das Modul selbst (Siehe Abbildung 26). Dies bedeutet, dass das Adaptive Grid zuerst das Modul platzieren möchte, und im Anschluss den «Close-Button». Dies führte zu Layout-Problemen. Die Lösung hierfür war, dass in jedem Modul ein Close-Button integriert wurde und die Dashboard Komponente den Modulen die Funktion für das Schliessen des Elementes mitgibt. (Siehe Abbildung 27 das Attribut «removeModuleFn»).

```
if (eL !== undefined && eL.nodeType === 1) {
```

Abbildung 28: Überprüfung des Nodetypes

Des Weiteren gab es verschiedene HTML-Elemente wie Breaking-Points und Kommentare, welche wir aus der Layoutgenerierung ausschliessen mussten. Deshalb war es wichtig eine Überprüfung des Nodetypes einzuführen (Siehe Abbildung 28).

5.6 Ausfallverhalten implementieren

Ein weiterer Vorteil der Adaptive Grid Implementation über eine erweiterte HTML-Element Klasse ist, dass alle Kinderelemente des Adaptive Grids immer dargestellt werden. Falls der Algorithmus einen Fehler aufwirft und/oder durch einen unvorhergesehenen Grund unterbricht, werden die Kinderelemente immer noch dargestellt. Sie werden jedoch keine Sortierung aufweisen. Jedoch kann die Verfügbarkeit aller Kinderelemente durch diese Technik garantiert werden. Des Weiteren haben wir ein Attribut hinzugefügt, welches die Deaktivierung und Aktivierung des Adaptive Grids zulässt. Somit kann auch für bestimmte Situationen das Adaptive Grid pragmatisch deaktiviert werden.

5.7 Integration der Regeln

Um das Adaptive Grid möglichst flexibel zu gestalten, haben wir das Konzept der "Regel" eingeführt. Dies ermöglicht es dem Entwickler, die Platzierung bestimmter Elemente im Adaptive Grid zu steuern.

Im Rahmen der IP5 Arbeit konnten wir das Regel-Konzept nicht vollständig implementieren, haben jedoch eine erste Version bereits für die Enablerr Plattform integriert.

Wie bereits in vorherigen Kapiteln beschrieben, ist die Universalität unseres Adaptiv Grids eine zentrale Stärke, welche wir auch bei unseren Regeln weiterführen wollen. Dies erreichen wir, in dem wir als Steuerelement das HTML ‚class‘ Attribut verwenden. Dieses Attribut wird von allen Browsern unterstützt und ermöglicht es uns ein sehr flexibles Regelwerk zu implementieren.

In der Instanziierung des Adaptive Grids, können dem Konstruktor zwei JSON-Objekte mitgegeben werden. Wir nennen diese beiden JSON-Objekte "areaMap" und "ruleSet"

5.7.1 ruleSet

Im „ruleSet“ kann konfiguriert werden, welche Klassen zu welchen Areas zugeordnet werden. In diesem Beispiel sehen wir, dass alle Elemente mit der CSS-Klasse „right-top-class“ der Area „right-top“ zugeordnet sind.

```
const ruleSet = {
  "positioning": [
    {
      "area": "right-top",
      "affected": "right-top-class"
    },
  ],
};
```

Abbildung 29: Beispiel Ruleset

5.7.2 areaMap

Die areaMap ist ein Register, dass in spezielle Sektionen innerhalb des Adaptive Grids konfiguriert werden kann.

```
const areaMap = {
  "left-top": {"x": 0, "y": 0},
  "left-bottom": {"x": 0, "y": 'len'},
  "right-top": {"x": 'len', "y": 0},
  "right-bottom": {"x": 'len', "y": 'len'},
  "new-module-button": {"x": 0, "y": 0}
}
```

Abbildung 30: Beispiel areaMap

Der Key dient als eindeutiger Schlüssel um die Sektionen identifizieren zu können. Jedem Key ist ein Objekt mit den x und y Koordinaten zugeordnet. Durch die Koordinaten, weiss das Adaptive Grid welche Bereiche im Adaptive Grid für diese Sektionen reserviert werden müssen.

In der Abbildung 29 ist ersichtlich, dass der Area „right-top“ die Koordinaten x: „len“ und „y“: 0 zugewiesen wurden.

Das Wort „len“ ist ein Platzhalter für die letzte Zeile oder Spalte innerhalb des Adaptive Grid. In diesem Fall sollen alle Elemente, mit der Klasse „right-top-class“ oben rechts platziert werden.

5.7.3 Verarbeitungsprozess

Im Prozessierungsablauf des Adaptive Grids werden Elemente, die von einem Regelwerk betroffen sind, priorisiert. Dies bedeutet, dass für diese Elemente, Zellen reserviert werden müssen. Dafür haben wir ein neues Register im Adaptive Grid erstellt mit dem Namen „Reservations“. Nach dem die areaMap und das ruleSet dem Adaptive Grid per Konstruktor mitgeteilt wurden, werden alle Kinder Elemente in Proxy Zellen gehüllt. In dieser Funktion findet bereits die Priorisierung der Elemente statt. Für jedes Element wird geprüft, ob das Element eine CSS-Klasse enthält, welches im ruleSet aufgeführt ist. Elemente, für die diese Eigenschaft nicht zutrifft, landen wie bereits beschrieben im CellStorage. Ist ein Element Teilmenge eines Regelwerks, wird die entsprechende Regel aus dem ruleSet extrahiert. Basierend auf der extrahierten Regel, werden im areaMap die Koordinaten ausgelesen und der Proxy Zelle zugewiesen. Dabei wird geprüft, ob das Wort „len“ in den Koordinaten enthalten ist. Ist dies der Fall weiss das Adaptive Grid, dass dieses Element auf der x oder y-Achse an letzter Position platziert werden muss. In so einem Fall, muss das Adaptive Grid prüfen wie hoch oder bereits das Element ist und dementsprechend wie viele Zeilen oder Spalten das Element im Adaptive Grid einnimmt. Diese Informationen sind zu diesem Zeitpunkt bereits bekannt. Ist die row oder colSpan addiert mit der Position grösser

als die korrespondierende Grenze auf der entsprechenden Achse, müssen die Koordinaten neu gesetzt werden. In so einem Fall, subtrahiert das Adaptive Grid die row oder colSpan den entsprechenden Koordinaten. Dieses System ermöglicht es uns die Entstehung unnötiger Lücken zu verhindern. Wenn mit den aktuellen Dimensionen kein Platz für eine weitere Reservation vorhanden ist, die Startposition aber frei wäre, wird das Raster automatisch erweitert.

5.7.4 Optimierung der Iteration

Mit dem Reservationssystem haben wir ein neues System eingeführt, wie das Adaptive Grid Zellen als besetzt markieren kann. Um dies zu ermöglichen, gibt es neu ein Register. Im Gegensatz zum zweidimensionalen Array, welches wir verwenden, um die restlichen Elemente zu platzieren, werden im Register nur Informationen für effektive Reservationen hinzugefügt. Dadurch sparen wir nicht nur Speicher, sondern haben auch einen Performance Boost bei der Prüfung ob Zellen frei sind.

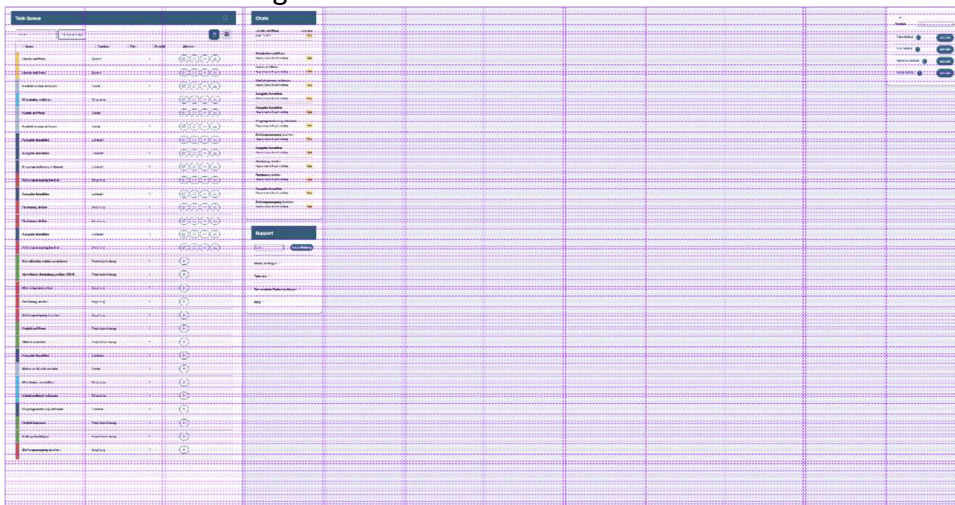


Abbildung 31: Unterteilung eines Layouts in Zellen

In diesem Beispiel stellen wir den ursprünglichen Ansatz mit dem neuen Register-Ansatz gegenüber.

schneller durchlaufen werden kann. Der Vorteil des neuen Ansatzes ist, dass er eine geringere Speicherbelastung hat, da er weniger Platz benötigt und somit auch schneller sein kann. Ein Nachteil ist jedoch, dass es schwieriger sein kann, bestimmte Elemente im Array zu finden, da es keine zweite Dimension gibt, um sie zu organisieren. Da für das Adaptive Grid die Performance allerdings eine höhere Gewichtung als die Komplexität hat, bleiben wir bei dem Register-Konzept. In einer ersten Umsetzungsiteration wurden nur Reservierungen über das Register verarbeitet, während alle anderen Elemente weiterhin über ein zweidimensionales Array verarbeitet wurden. Wir waren jedoch unzufrieden mit dieser Hybridlösung und haben unseren Density-Algorithmus daher komplett überarbeitet. Das zweidimensionale Array, das zuvor für die Platzierung der Elemente verwendet wurde, wurde entfernt. Jetzt werden alle Elemente über dasselbe Register gesteuert wie die Reservierungen. Diese Veränderung hat nicht nur die Leistung unserer Anwendung deutlich verbessert, sondern auch ermöglicht, etwa die Hälfte des gesamten Codes zu entfernen und die Übersicht und Verständlichkeit des Codes zu verbessern.

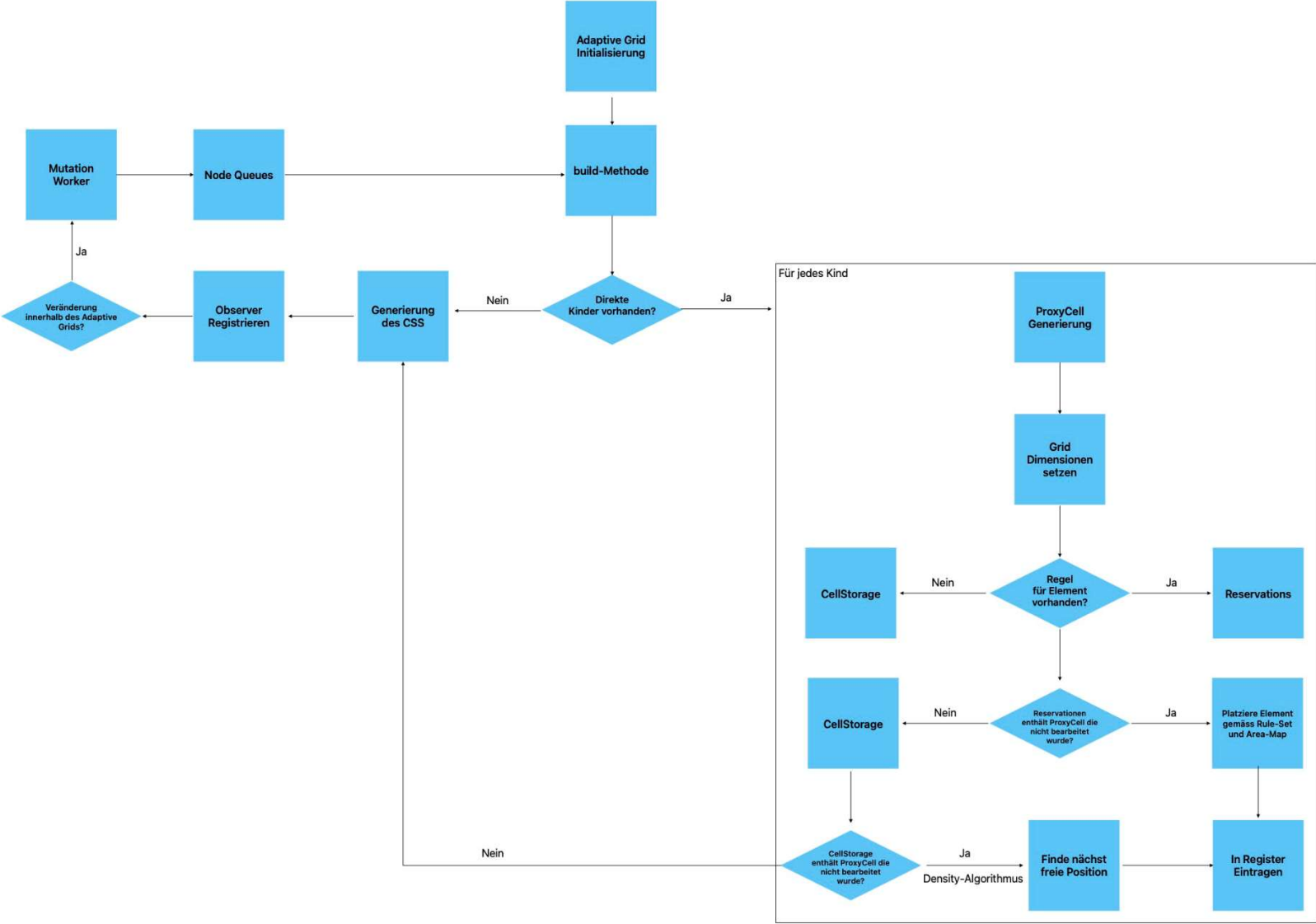


Abbildung 34: Der Prozessablauf innerhalb des Adaptive Grids resultiert aus Implementierungen und Optimierungen, die wir im Vergleich zum Konzept durchgeführt haben.

6 Evaluierung

6.1 Vorgehensweise

Die Evaluierung des Adaptive Grids wurde in verschiedene Teile unterteilt. Die Unterteilung ist wie folgt:

- **Evaluierung des Adaptive Grids in Enablerr vs. Svelte Grid in Enablerr**
Wir haben 8 Leute an die Enablerr Plattform hingesetzt und eine fixe Aufgabe erteilt im Dashboard. Diese Aufgabe haben Sie einmal durchgeführt im Svelte Grid und einmal im Adaptive Grid. Anschliessend wurde eine Konversation mit diesen Personen durchgeführt, um zu evaluieren, welche Lösung sich besser angefühlt hat.
- **Evaluierung des Adaptive Grids am Playground**
Des Weiteren haben wir eine Evaluierung am Adaptive Grid Playground durchgeführt. Hiermit probierten wir möglichst viele verschiedene Elementkombinationen aufzuführen und evaluierten die Darstellung auf Layout und Funktionalität.
- **Evaluierung des Adaptive Grids bzgl. Einsatzpotential**
Des Weiteren führten wir eine Evaluation durch, um herauszufinden, welche Einsatz- und Weiterentwicklungsmöglichkeiten für das Adaptive Grid bestehen.

6.2 Risiken in der Evaluierung

Eines der ersten Risiken bestand darin, dass die Evaluierung im Enablerr nicht ausreichend repräsentativ sein könnte. Aufgrund von Zeitbeschränkungen konnten wir leider keine Tests mehr mit einer großen Anzahl von Teilnehmern durchführen. Das Risiko bestand darin, dass alle Testpersonen einen einseitigen Hintergrund hatten. Um dies zu vermeiden, haben wir uns dafür entschieden, eine möglichst diverse Gruppe von Testpersonen einzuladen. Wir haben sowohl Männer als auch Frauen eingeladen, sowie Studenten und Personen im Alter von 30-40, 40-50 und 50-63. Wir haben auch Personen ausgewählt, die im Job ERP-Systeme verwenden müssen. Die Häufigkeit der Interaktion mit der ERP-Software variiert jedoch von täglich bis zu einmal pro Woche für etwa 30 Minuten.

Ein weiteres Risiko bei der Evaluierung von Enablerr bestand darin, dass die Implementierung auf einem Dashboard durchgeführt wurde, das zu diesem Zeitpunkt nur vier Elemente enthielt. Weitere Elemente waren vom Entwicklungsteam noch nicht implementiert worden. Mit nur vier Elementen im Dashboard ist es schwierig, eine aussagekräftige Evaluierung des Adaptive Grids durchzuführen, da viele der Funktionen des Adaptive Grids bei einer geringen Anzahl von Kinderelementen nicht genutzt werden können. Daher haben wir die Evaluierung des Adaptive Grids auf den Playground ausgeweitet.

6.3 Zielgruppe analysieren

Die Zielgruppe von Enablerr ist ziemlich weitläufig, da jegliche Firmen/Personen ein ERP-System benötigen. Somit schliesst dies alle Personen mit ein, welche einen Job besitzen und mindestens Teilzeit im Büro arbeiten und ein ERP-System benützen.

Eine weitere Zielgruppe des Adaptive Grids sind Entwickler. Durch das Adaptive Grid kann sich der Entwickler Stunden an Zeit sparen, da jegliche verschiedene Eventualitäten für die Darstellung der verschiedenen Elemente nicht entwickelt werden müssen.

6.4 Bedienbarkeit des Enablerr

6.4.1 Auswertung

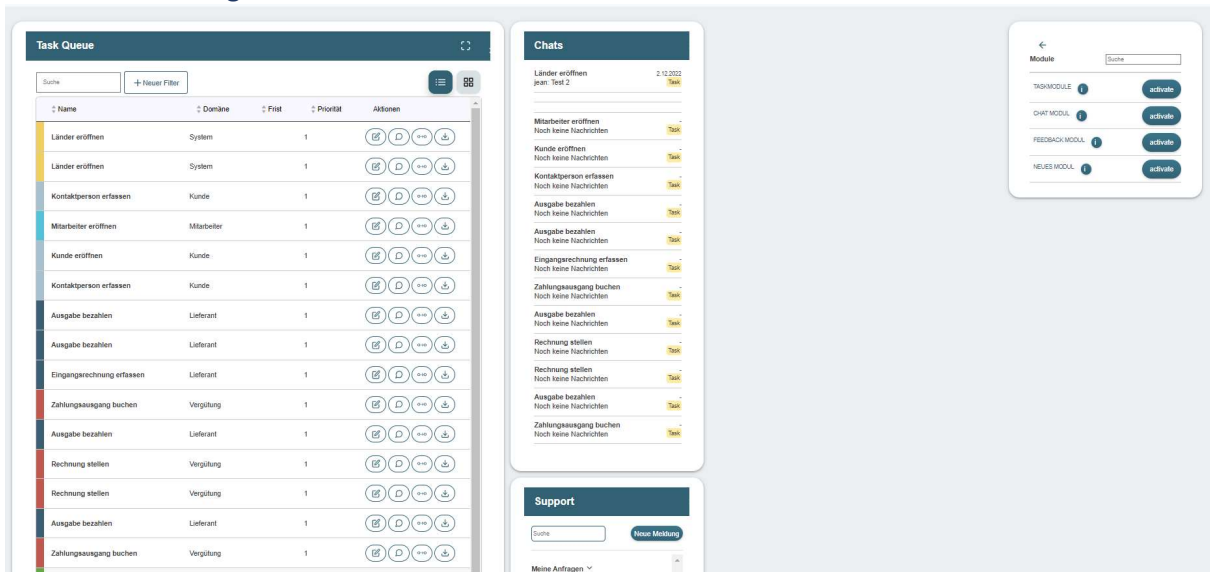


Abbildung 35: Enablerr Dashboard mit implementiertem Adaptive Grid

Wie Sie in Abbildung 35 erkennen können besteht das Dashboard aus gesamthaft 4 Elementen. Das Hinzufügen und Entfernen dieser Elemente war Teil der Aufgabe in der Evaluierung. Des Weiteren mussten Sie einen Task öffnen und eine neue Supportmeldung schicken.

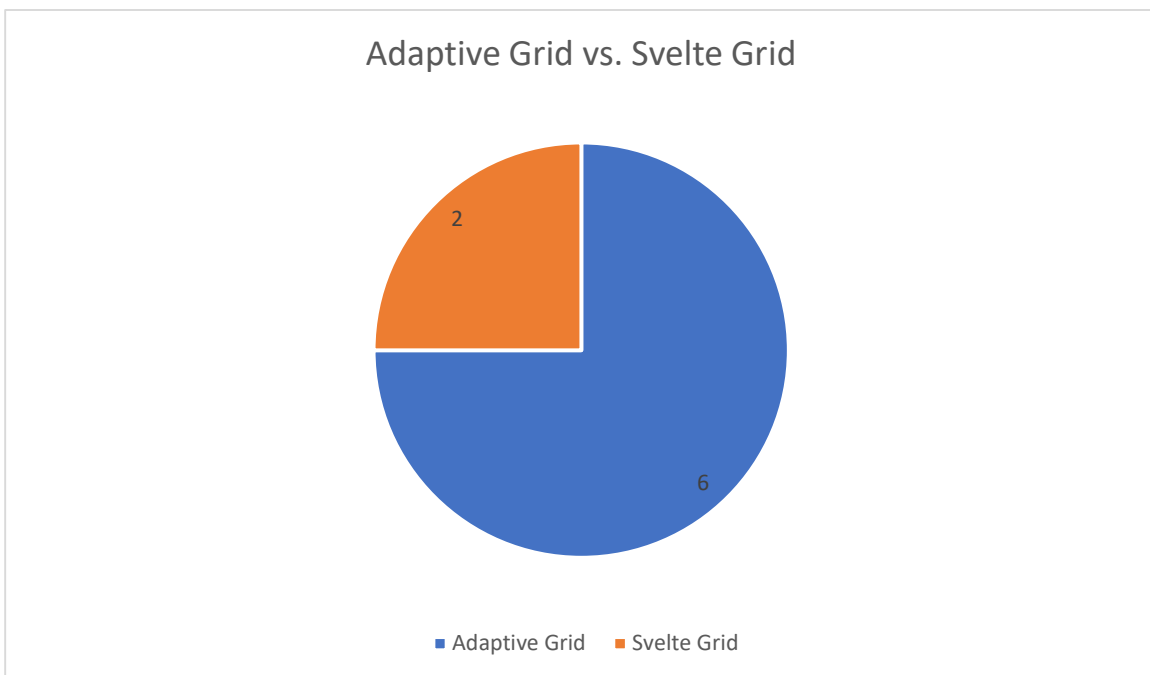


Abbildung 36: Adaptive Grid vs. Svelte Grid: Entscheidung der Testpersonen

Wie Sie in Abbildung 36 erkennen können, haben sich sechs unserer Testpersonen darüber geäußert, dass sie die Adaptive Grid Implementation bevorzugen. Als Hauptgrund wurde die andere Anordnung des «Module Hinzufügen Elementes» (Siehe Abbildung 35 oben rechts) benannt. Im Svelte Grid wurde dieses Modul mit dem Rest der Elemente nebeneinander angezeigt. Jedoch ist im Adaptive Grid dieses Element immer an die Position oben rechts gegliedert. Dies führte dazu, dass weniger Verwirrung entstand beim Hinzufügen und

Entfernen von Elementen, da das «Module Hinzufügen Element» immer an derselben Stelle blieb bei der Adaptive Grid Implementation.

Die zwei Personen, welche die Svelte-Grid Implementation bevorzugten, haben als Grund die Drag & Drop Funktion genannt. Die zwei Testpersonen bevorzugten eine eigene Konfiguration des Dashboards über eine vorhergesehene.

6.4.2 Verbesserungsvorschläge

Als erstes würden wir empfehlen, die Adaptive Grid Implementation nochmals zu evaluieren, wenn mehr Elemente im Dashboard hinzugefügt werden können. Des Weiteren könnte man die Drag & Drop Funktion miteinbauen in das Adaptive Grid. Die genaue Positionierung eines jeden Elementes ist möglich und die Reservation eines bestimmten Platzes im Adaptive Grid ist ebenfalls möglich. Somit müsste man die «Drag & Drop» Events noch verknüpfen mit der Umpositionierung und der Reservierung.

6.5 Technische Umsetzung

6.5.1 Verbesserungsvorschläge

Eine weitere Möglichkeit, die Benutzerfreundlichkeit zu erhöhen, besteht darin, eine Ladeanimation einzubauen, wenn das Adaptive Grid neu generiert wird. Dadurch kann der Nutzer erkennen, dass das System aktiv arbeitet, und die Wartezeit wird für den Nutzer angenehmer gestaltet.

Aufgrund von Zeitbeschränkungen haben wir es leider nicht geschafft, alle Regeln vollständig umzusetzen. Dies führt derzeit zu eingeschränkten Möglichkeiten in Bezug auf die Anwendung des Regel-Features. Um die Funktionalität zu verbessern, besteht die Möglichkeit, diesen Bereich weiter auszubauen.

6.6 Integration als Universal Grid

6.6.1 Auswertung

Im Playground konnten wir das Adaptive Grid auf verschiedene Elementkombinationen testen und feststellen, dass der Density-Algorithmus alle Eventualitäten abdeckt. Durch die Verwendung der areaMap und des ruleSets konnten wir die Umgebung noch weiter an unsere Anforderungen anpassen. Dadurch konnten wir das Adaptive Grid als funktionsfähig erklären und uns auf weitere Entwicklungen konzentrieren. Es ist auch wichtig zu erwähnen, dass durch die aktuelle Umsetzung ein solides Fundament für zukünftige Entwicklungen geschaffen wurde.

6.6.2 Verbesserungsvorschläge

Benutzerkontext: Eine mögliche Weiterentwicklung wäre es, das UI anzupassen gemäss dem Benutzerkontext. Man könnte z.B. bestimmte Elemente an einer anderen Stelle anzeigen basierend auf der Benützungshäufigkeit des Elementes oder basierend auf das Alter des Benutzers.

Inhaltsordnung: Bis jetzt überprüft das Adaptive Grid den Inhalt der Kinderelemente nicht. Jedoch wurde das Adaptive Grid absichtlich so entwickelt, dass die Entscheidung der Endposition des Elementes möglichst einfach ausgeweitet werden kann. Dies bedeutet, dass z.B. ein Algorithmus entwickelt werden könnte, welcher den Inhalt der Kinderelemente analysiert und diese nach Zugehörigkeit anordnet. Des Weiteren könnte z.B. ein White Space oder ein Divider automatisch eingesetzt werden an Stellen, bei denen sich die Themen ändern.

Ordnung bestimmter HTML-Elemente: Des Weiteren könnte man den Adaptive Grid dazu entwickeln, dass er bestimmte HTML-Elemente immer an einer bestimmten Stelle anordnet.

Z.B. Buttons immer unten rechts, Containers immer im Center, etc. Mit der aktuellen Umsetzung ist dies nur durch Klassen möglich. Aufgrund der Flexibilität unseres Regelsystems sollte eine Weiterentwicklung dieser Funktion kein Hindernis darstellen.

Interaktive Ordnung: Im Kapitel 6.4.2 haben wir schon erwähnt, dass eine Drag & Drop Funktion eine mögliche Weiterentwicklung wäre. Des Weiteren würden wir jedoch vorschlagen, dass z.B. das Anheften eines Elementes an einen Rand interaktiv gestaltet wird. Somit kann der Benutzer einen bestimmten Einfluss nehmen auf das Layout an den vom Benutzer gewünschten Stellen.

Mobile Layouts: Eine weitere Funktion, die man implementieren könnte, wäre die mobile Adaptivität des Layouts. Das Adaptive Grid könnte auf Viewport Wechsel hören und dementsprechend das Layout basierend auf Regeln anpassen.

6.7 Evaluierung der Einsatzmöglichkeiten

6.7.1 Enablerr Tasks

Die Implementierung eines neuen Systems bietet oft eine Herausforderung, insbesondere wenn es darum geht, eine solide Grundlage für zukünftige Erweiterungen und Verbesserungen zu schaffen. In diesem Fall haben wir ein Regelsystem implementiert, das zwar noch nicht vollständig abgeschlossen ist, jedoch ein solides Fundament für die Enablerr Tasks bietet, auf dem weitere Funktionen aufgebaut werden können.

Das System verfügt über eine robuste Datenstruktur, die es ermöglicht, Regeln einfach zu implementieren, zu verwalten und nachzuverfolgen. Mit der aktuellen Implementierung, wie wir sie im Kapitel 5.7 beschreiben, haben wir sichergestellt, dass das System skalierbar und zukunftssicher ist.

Obwohl das Regelsystem derzeit noch begrenzte Funktionalitäten bietet, bietet es dennoch eine solide Grundlage für zukünftige Erweiterungen.

In der Zukunft planen wir, das Regelsystem weiter auszubauen, um es noch leistungsfähiger und nützlicher für Benutzer des Adaptive Grids zu machen. Wir sind zuversichtlich, dass das solide Fundament, das wir jetzt geschaffen haben, uns dabei helfen wird, diese Ziele zu erreichen.

6.7.2 Automatisch generiertes UI

Das Adaptive Grid eignet sich im Einsatz ideal an allen Stellen, wo ein UI automatisch generiert wird. Denn durch das Adaptive Grid kann man auf eine komfortable und einfache Weise bestimmte Elemente an bestimmten Orten positionieren und der Rest wird gemäss dem Density Algorithmus angeordnet. Dies spart dem Entwickler Zeit und gibt dem Entwickler des Weiteren ein mächtigeres Tool wie ein «normales» Grid oder z.B. eine «display:flex» Instanz.

6.7.3 Dynamisches UI

Ein idealer Einsatzbereich für das Adaptive Grid sind Orte, an denen das UI durch dynamische Kinderelemente dynamisch verändert werden kann. Ein Beispiel hierfür ist eine Webseite, die Bewertungen mit unterschiedlichen Größen anzeigt und sich durch neue Bewertungen anpassen kann. Mit dem Adaptive Grid kann man diese auf ansprechende Weise anordnen, ohne viel Zeit in das Design zu investieren.

7 Fazit

Während unserer Recherche haben wir festgestellt, dass die meisten Arbeiten zu Adaptive-UI sich hauptsächlich mit der Verarbeitung von Benutzer- und Systemkontext beschäftigen. Es existieren jedoch nur wenige Informationen und Arbeiten zur spezifischen Generierung von Layouts und UI-Komponenten im Zusammenhang mit Adaptive-UI. Unser Adaptive Grid füllt diese Lücke und bietet die Möglichkeit, ein zur Laufzeit interaktives Gridsystem zu generieren. Das Konzept wurde erfolgreich umgesetzt und durch Ergänzungen verbessert. Unser Produkt ist skalierbar und performant, da jeder Benutzer und Entwickler es durch den Einsatz von Rulesets erweitern kann. Nach Durchführung von Evaluierungen haben wir positive Rückmeldungen erhalten. Es ist jedoch wichtig zu erwähnen, dass wir eine Reevaluierung des Adaptive Grids im Use-Case des Enablerr vorschlagen, wenn mehr Komponenten im Dashboard vorhanden sind. Das Adaptive Grid bietet ein solides Fundament und Grundgerüst, das eine Vielzahl von Anforderungen abdeckt. Unser npm-Paket wurde bereits veröffentlicht und hat in der ersten Woche über 200 Downloads erhalten. Insbesondere ist das Potential, die Ausbaumöglichkeiten und die Einsatzmöglichkeiten von Adaptive Grid interessant.

8 Literaturverzeichnis

- [1] pier4all AG, „Enablerr,“ 2022. [Online]. Available: <https://enablerr.ch>.
- [2] J. M. E. Belo, M. Lystbaek, A. M. Feit, K. Pfeuffer, P. Kán, A. Oulasvirta und J. Gronbaek, „AUIT - The Adaptive User Interfaces Toolkit for Designing XR Applications,“ in *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*, New York, 2022.
- [3] M. Dubiel, B. A. Yilma, K. Latifzadeh und L. Leiva, „A Contextual Framework for Adaptive User Interfaces: Modelling the Interaction Environment,“ University of Luxembourg, Luxembourg, 2022.
- [4] M. Sadowski, „Semantic-Based Design of an Adaptive UI,“ in *Communications in Computer and Information Science, Volume 1625*, OSTIS, 2022, pp. 176 - 202.
- [5] A. Braham, M. Khemaja, F. Buendía und F. Gargouri, „Towards a Model-Driven Ontology-Based Architecture for Generating Adaptive User Interfaces,“ in *Lecture Notes in Network and Systems book Volume 483*, 2022.
- [6] Z. Ruan, Y. Fukazawa und J. Shirogane, „Automatic generation of user-sensitive and application-sensitive self-adapted UI system for smartphone applications,“ in *2022 International congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, Ankara, 2022.
- [7] M. H. Miraz, A. Maaruf und E. Peter S., „Adaptive user interfaces and universal usability through plasticity of user interface design,“ 2021.
- [8] K. Pfeuffer, Y. Abdrabou, A. Esteves, R. Rivu, Y. Abdelrahman, S. Meitner, A. Saadi und F. Alt, „ARtention: A design space for gaze-adaptive user interfaces in augmented reality,“ in *Computers & Graphics Volume 95*, Science Direct, 2021, pp. 1 - 12.
- [9] K. Todi, G. Bailly, L. Leiva und A. Oulasvirta, „Adapting User Interfaces with Model-based Reinforcement Learning,“ in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, 2021.
- [10] T. Horbinski, P. Cybulski und B. Medynska-Guilij, „Web Map Effectiveness in the Responsive Context of the Graphical User Interface,“ ISPRS, 2021.
- [11] E. Yigitbas, I. Jovanovikj, K. Biermeier, S. Sauer und G. Engels, „Integrated model-driven development of self-adaptive user interfaces,“ in *Software Systems Modeling 19*, 2020, pp. 1057-1081.
- [12] E. Yigitbas, „Model-driven engineering and usability evaluation of self-adaptive user interfaces,“ in *ACM SIGWEB Newsletter, Issue Autumn 2020*, 2020, pp. 1 - 4.
- [13] A. Taqdir, J. Hussain, M. B. Amin, U. Akhtar, W. A. Khan, S. Lee, B. H. Kang, M. Hussain, M. Afzal, H. W. Yu, U. U. Rehman, H.-S. Han, J. Y. Choi und A. Jamshed, „The Intelligent Medical Platform: A Novel Dialogue-Based Platform for Health-Care Services,“ in *Computer, Vol. 53*, 2020, pp. 35-45.
- [14] A. Wijesundara, „Engineering Privacy-aware Smart Home Environments,“ in *Proceedings of the 12th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, 2020.

- [15] E. Yigitbas, K. Josifovska, I. Jovanovikj, F. Kalinci, A. Anjorin und G. Engels, „Component-based development of adaptive user interfaces,“ in *EICS '19: Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, 2019.
- [16] N. Rathnayake, D. Meedeniya, I. Perera und A. Welivita, „A Framework for Adaptive User Interface Generation based on User Behavioural Patterns,“ in *2019 Moratuwa Engineering Research Conference (MERCon)*, 2019, 2019.
- [17] B. B A und S. H B, „TI - Adaptive User Interface of Learning Management Systems for Education 4.0: A Research Perspective,“ *Journal of Physics: Conference Series* , Bd. 1235, 2019.
- [18] S. Kolekar, R. Pai und M. Pai, „Rule based adaptive user interface for adaptive E-learning system,“ in *Education and Information Technologies 24*, 2019, pp. 613-641.
- [19] M. Heck, J. Edinger und C. Becker, „Gaze-based Product Filtering: A System for Creating Adaptive User Interfaces to Personalize Stateless Point-of-Sale Machines,“ in *UIST '19: The Adjunct Publication of the 32nd Annual ACM Symposium on User Interface Software and Technology*, 2019.
- [20] J. Hussain, A. U. Hassan, H. S. M. Bilal, R. Ali, M. Afzal, S. Hussain, J. Bang, O. Banos und S. Lee, „Model-based adaptive user interface based on context and user experience evaluation,“ in *Journal on Multimodal User Interfaces 12*, 2018, pp. 1 - 16.
- [21] M. Nawaz, L. Motiwalla und A. Deokar, „Adaptive User Interface for a Personalized Mobile Banking App: Extended Abstract,“ in *UMAP '18: Adjunct Publication of the 26th Conference on User Modeling, Adaptation and Personalization*, 2018.
- [22] L. Zhang, „Object-oriented User Interface Customization Framework: Customizing Complex User Interfaces to Improve Usability and User Performance,“ Perdue University, Indiana, 2018.
- [23] H.-Z. Tan, W. Zhao und H.-H. Shen, „Adaptive user interface optimization for multi-screen based on machine learning,“ in *2018 IEEE 22nd International Conference on Computer Supported Cooperative Work in Design ((CSCWD))*, 2018.
- [24] H. Soh, S. Sanner, M. White und G. Jamieson, „Deep Sequential Recommendation for Personalized Adaptive User Interfaces,“ in *IUI '17: Proceedings of the 22nd International Conference on Intelligent User Interfaces*, 2017.
- [25] K. Gajos und K. Chauncey, „The Influence of Personality Traits and Cognitive Load on the Use of Adaptive User Interfaces,“ in *IUI '17: Proceedings of the 22nd International Conference on Intelligent User Interfaces*, 2017.
- [26] E. Yigitbas, S. Grün, S. Sauer und G. Engels, „Model-Driven Context Management for Self-adaptive User Interfaces,“ in *Lecture Notes in Computer Science Volume 10586*, 2017.
- [27] D. Hariyanto und T. Kohler, „An Adaptive User Interface for an E-learning System by Accommodating Learning Style and Initial Knowledge,“ in *ICTVT 2017*, 2017.
- [28] S. Bouzit, C. Gaele, J. Coutaz, D. Chene, E. Petit und J. Vanderdonck, „Design Space Exploration of Adaptive User Interfaces,“ 2017.
- [29] „FAST,“ Microsoft, 2022. [Online]. Available: <https://www.fast.design/>.
- [30] „Flutter Documentation,“ Goj, 2022. [Online]. Available: <https://docs.flutter.dev/development/ui/layout/adaptive-responsive>.

- [31] „mdn Web Docs,“ Mozilla, [Online]. Available: <https://developer.mozilla.org/en-US/docs/Web/API/MutationObserver>. [Zugriff am 2022].
- [32] J. Yablonski, „Laws of UX,“ 2022. [Online]. Available: <https://lawsofux.com/>. [Zugriff am 2022].
- [33] „Github,“ Microsoft, [Online]. Available: <https://github.com/>. [Zugriff am 2022].
- [34] „Svelte,“ [Online]. Available: <https://svelte.dev/>. [Zugriff am 2022].
- [35] V. Araqelyan, „Svelte Grid,“ [Online]. Available: <https://svelte-grid.vercel.app/>. [Zugriff am 2022].
- [36] Svelte, „Svelte Development,“ [Online]. Available: <https://svelte.dev/tutorial/onmount>. [Zugriff am 2022].

9 Abbildungsverzeichnis

Abbildung 1: Übersicht Planung des Projektes	5
Abbildung 2: Filterung der wissenschaftlichen Recherche – Relevante und irrelevante Papers	15
Abbildung 3: Erwähnte Technologien	16
Abbildung 4: Ablauf Anordnung der Elemente	18
Abbildung 5: Vordefinierte Elementgrößen	19
Abbildung 6: Darstellung Empty Space	20
Abbildung 7: Mögliche Anordnung von Elementen	21
Abbildung 8: Adaptive Grid Model	22
Abbildung 9: ProxCells mit elementspezifischen Werten	23
Abbildung 10: Ordnungsprozess	23
Abbildung 11: Dashboard Enablerr	25
Abbildung 12: Adaptive Grid Playground	26
Abbildung 13: Integration des Adaptive Grids in eine HTML Seite	27
Abbildung 14: Der State des <p> Tags verändert sich. Dadurch wird der State Observer aktiviert. Da wir nur direkte Nachkommen des Adaptive Grid Elementes verarbeiten, muss im DOM der Baum so weit nach oben geklettert werden, bis ein Knoten gefunden wird, welcher der direkte Nachkomme des Adaptive Grids ist. Dafür haben wir die Methode findDirectSuccessor erstellt. Das gefundene Element wird anschliessend vom Observer in die Node-Pipeline hinzugefügt	28
Abbildung 15: Problemdarstellung des kleinsten Elementes und der Zeilenhöhe	30
Abbildung 16: Visualisierung des Grids wenn die Zeilenhöhe fix gesetzt wird	30
Abbildung 17: Fehlerhafter Generierung der Containerbreite und der Element Höhen durch den row und col-gap.	31
Abbildung 18: Korrektur der Gesamtbreite mit Hilfe der calc() Funktion	31
Abbildung 19: Korrekte Ausgabe der Höhe aller Elemente	31
Abbildung 20: Nach den Korrekturen aus der Abbildung 18 und Abbildung 19, wird das Grid Layout korrekt generiert und der Zeilen und Spaltenabstand ist korrekt definiert.	32
Abbildung 21: Initialisierung des Adaptive Grid mit col-span und row-span	32
Abbildung 22: Visualisierung der Abstände nach der Layout Generierung	32
Abbildung 23: Integration der Webkomponente per Skript Tag	33

Abbildung 24: Platzierung des <adaptive-grid> HTML-Elementes	33
Abbildung 25: Code der Dashboard Komponente vor Integration	34
Abbildung 26: Code der Dashboard Komponente nach Integration	34
Abbildung 27: Überprüfung des Nodetypes	35
Abbildung 28: Beispiel Ruleset	36
Abbildung 29: Beispiel areaMap	36
Abbildung 30: Unterteilung eines Layouts in Zellen	37
Abbildung 31: Darstellung des zweidimensionalen Arrays in der Browser Konsole.	38
Abbildung 32: Register Array, welches nur noch ein Minimum an Daten enthält.	38
Abbildung 33: Der Prozessablauf innerhalb des Adaptive Grids resultiert aus Implementierungen und Optimierungen, die wir im Vergleich zum Konzept durchgeführt haben.....	40
Abbildung : Enablerr Dashboard mit implementiertem Adaptive Grid	42
Abbildung : Adaptive Grid vs. Svelte Grid: Entscheidung der Testpersonen	42

10 Tabellenverzeichnis

Table 1: Wissenschaftliche Recherche Adaptives UI.....	8
--	---