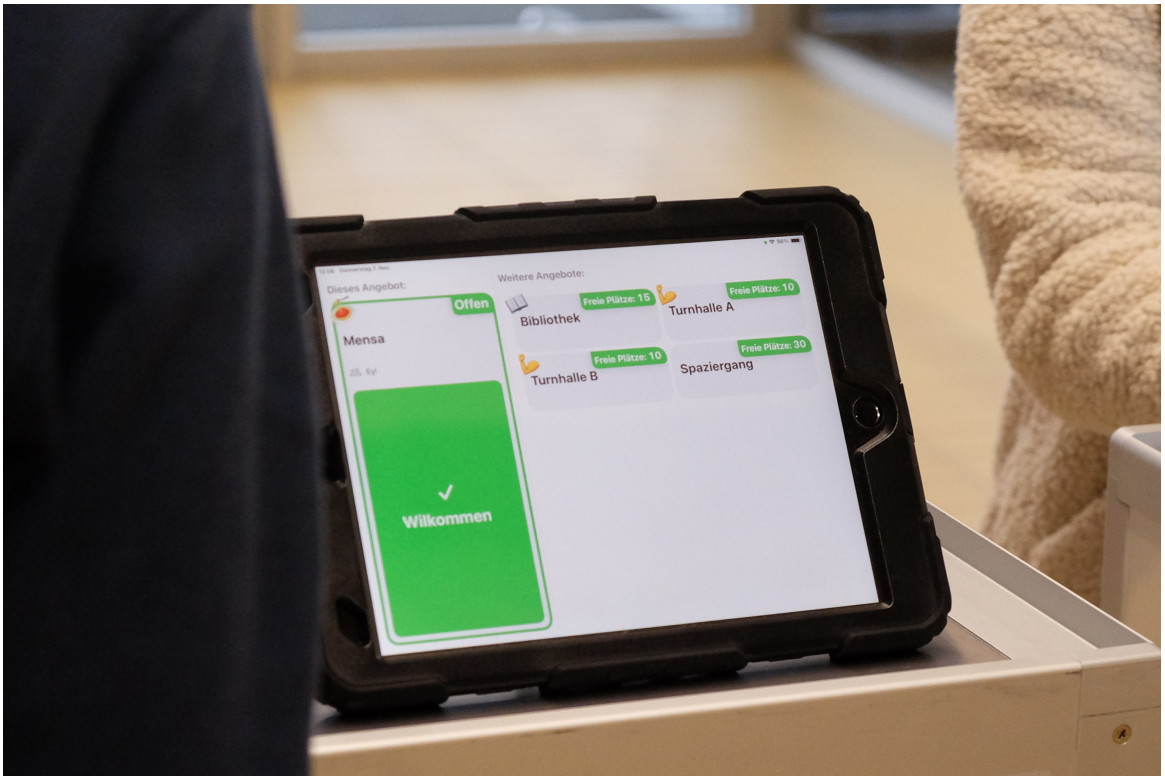
Maturitätsarbeit an der Kantonsschule Zürich Nord

ORBITS: Entwicklung eines digitalen Organisations-Systems für eine Zürcher Tagesschule

Erik Schnell, M6f

Betreuungsperson: **Lukas Zuberbühler**



Zürich, November 2024

Zusammenfassung

Diese Maturitätsarbeit befasst sich mit der Entwicklung einer digitalen Lösung zur Verwaltung der Mittagsbetreuung an der Schule Freilager in Zürich. Ziel war es, den administrativen Aufwand der Betreuungspersonen zu reduzieren. Hierfür wurde eine App entwickelt, die mithilfe von QR-Codes das Ein- und Auschecken der Kinder zu verschiedenen Angeboten ermöglicht. Bei der Auswahl der Hardware und der Entwicklung der Software lag ein besonderer Fokus auf dem Datenschutz. In enger Zusammenarbeit mit der Schule Freilager wurde die App über mehrere Wochen hinweg mit sechs Pilotklassen getestet. Die hier vorgeschlagene Lösung, die langfristig das bisherige manuelle System ablösen könnte, hat sich im Testzeitraum als vielversprechend erwiesen. Nach Weiterentwicklung des Systems ist es vorstellbar, dass es auch anderen Tagesschulen zugute kommen könnte. Dem System wurde der Name ORBITS gegeben: **Or**ganisation der **B**etreuung **i**n **T**ages**s**chulen.

Danksagung

Ich bin unglaublich glücklich mit ORBITS. Der gesamte Entwicklungsprozess bereitete mir enorm viel Freude.

Ich möchte Alex Müller, dem Schulleiter der Schule Freilager, danken, der mir die Möglichkeit gab, dieses Projekt umzusetzen. Ohne sein Vertrauen und seine Unterstützung wäre ORBITS nicht realisierbar gewesen. Danke, dass du mich mit einer Einladung zum Mittagessen aus der hauseigenen Mensa-Küche in die brandneue Schule locktest – wer hätte gedacht, dass aus einem Teller Pasta ein ganzes Projekt entsteht!
Ein gleich grosser Dank geht an Mark Voler, Schulleitungsassistent der Schule Freilager, der mir als Ansprechperson für das Projekt stets zur Seite stand. Danke für deine Begeisterung, deine Nerven und dafür, dass du meine E-Mail-Flut heldenhaft überlebtest – und natürlich auch dafür, dass du den Koch auf die Leiter gejagt hast, um den Router zu installieren. Ohne dich wäre das Projekt technisch und menschlich nur halb so gut geworden!

Vielen Dank auch an Lukas Zuberbühler, der sich bereit erklärte, mich bei dieser Arbeit zu betreuen. Ihr Feedback und Ihre Ratschläge waren eine riesige Hilfe, um die Arbeit erfolgreich voranzutreiben. Auch wenn Overleaf bei unseren Meetings mal wieder mein Projekt nicht kompilieren wollte!
Danke an Thomas Haupt, den Koreferenten meiner Arbeit, für deine Zeit und Mühe, meine Arbeit zu bewerten - und für deine Gitarrenskills.

Ein grosses Dankeschön geht an meine Mutter Christina Schnell – für ihre bedingungslose Liebe und Geduld. Auch wenn sie von Programmieren keinen blassen Schimmer hat, hielt sie mich mit ihrem Deutschwissen ordentlich auf Trab. Ohne dich wäre dieses Projekt halb so korrekt und doppelt so chaotisch geworden! Ebenfalls danke ich meiner Schwester Nora, die sich immer wieder bereitwillig als Versuchskaninchen zur Verfügung stellte – und dafür natürlich das coolste Konfetti von allen abstaubte!

Auch mein Vater Rémy Schnell verdient Dank – für seine ersten Inputs zum Projekt, für das gründliche Korrekturlesen und dafür, dass er die Freude an diesem Projekt mit mir teilte.
Vielen Dank an Christian Marfurt, der meine Arbeit mit Adleraugen durchlas und mir half, das kreative Chaos in eine saubere Struktur zu verwandeln.

Ein besonderes Dankeschön geht an Sidney Archer: für das Zensieren der Bilder, für die gegenseitige Motivation, an unseren Arbeiten weiterzumachen, und für das gemeinsame Jammern, wenn es mal nicht lief.

Ich danke den Betreuungspersonen der Schule Freilager, die mich so herzlich aufnahmen und mit ihrem tollen Feedback massgeblich zur Entwicklung des Systems beitrugen – und dafür, dass sie tapfer die schreckliche erste Version, inklusive Absturz, über sich ergehen liessen!
Auch den Schulleitungen und Betreuungspersonen der interviewten Schulen möchte ich herzlich danken. Ihre Offenheit und die Zeit, die sie sich für meine Fragen nahmen, unterstützten mich sehr in der Bedarfsanalyse.

Inhaltsverzeichnis

Abkürzungsverzeichnis

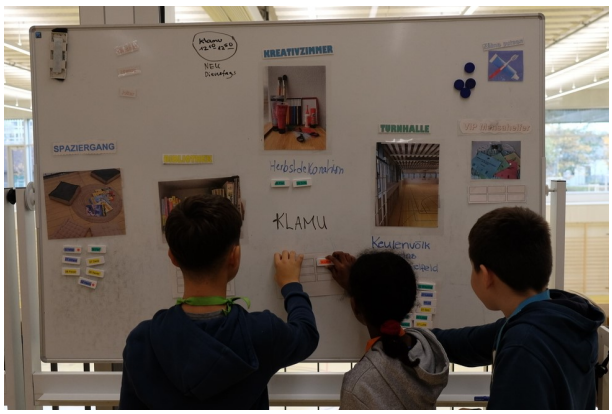
CPU	Central Processing Unit
CRUD	Create Read Update Delete
DocC	Documentation Compiler
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IDG	Gesetz über die Information und den Datenschutz
IDV	Verordnung über die Information und den Datenschutz
KITS	Kommunikations- und Informations-Technologien für die Schulen der Stadt Zürich
MVVM	Model–View–ViewModel
NFC	Near Field Communication
OIZ	Organisation und Informatik Zürich
ORM	Object-Relational Mapping
QR-Code	Quick Response Code
RFID	Radio-Frequency Identification
SQL	Structured Query Language
UI	User Interface
UX	User Experience
WLAN	Wireless Local-Area Network

1 Einleitung

Die Stadt Zürich stellt alle Volksschulen auf Tagesschulen um. Laut Schul- und Sportdepartement wird damit die Vereinbarkeit von Familie und Beruf erleichtert sowie Unterricht und Betreuung pädagogisch und organisatorisch zusammengeführt. 30 Schulen in der Stadt Zürich haben im Rahmen des Pilotprojekts «Tagesschule 2025» ihren Betrieb bereits erfolgreich umgestellt [1]. Dies geht mit weitgehenden strukturellen Änderungen einher. Insbesondere gehen die Kinder nunmehr grundsätzlich über Mittag nicht mehr nach Hause oder in den Hort, sondern können in der Schule versorgt und betreut werden.

1.1 Ausgangslage

Seit August 2022 ist auch die Primarschule Freilager in Zürich Albisrieden eine Tagesschule. Gemäss Auskunft des Schulleiters Alex Müller seien das Herz der neuen Mittagsbetreuung die Verpflegung in der Mensa und die offenen Betreuungsangebote. Die Kinder würden ihre 1.5-stündige Mittagspause frei gestalten und sich nach Belieben zwischen Angeboten bewegen können, die von «Mensa» über «Disco» bis hin zu «Turnhalle» reichten. Nach seiner Einschätzung würden die Kinder die Betreuungsangebote gut annehmen, jedoch stelle der administrative Aufwand für die Betreuungspersonen täglich eine Herausforderung dar. Eine von der Stadt Zürich erstellte Anwesenheitsliste zeige, welche Kinder für den jeweiligen Tag angemeldet seien. Um ihrer Aufsichtspflicht nachzukommen, werde jedes anwesende Kind jeden Mittag in dieser Excel-Liste vor der Mensa manuell abgehakt. Die Verwaltung dieses täglichen Ansturms von über 300 Kindern erfordere den Einsatz von zwei Betreuungspersonen, die über den gesamten Mittag hinweg ausschliesslich deren Namen im System abhaken müssten (vgl. Abb. 1a). Um einen Überblick über die Verteilung der Kinder zu behalten und jene zu finden, die noch nicht gegessen hätten, werde eine Magnettafel verwendet. Jedes Kind habe einen eigenen Magneten, den es bei jedem Wechsel des Angebots auf das entsprechende Feld der Tafel setzen müsse (vgl. Abb. 1b). Diese Magnete werden von einer Betreuungsperson vor der Mittagspause sortiert und bereitgelegt. Da einige Angebote begrenzte Plätze hätten und es gelte, das unerlaubte Verschieben der Magnete zu verhindern, überwachten täglich ein bis zwei Betreuungspersonen die Tafel vor Ort.
Dieser Ablauf sei derzeit unbefriedigend, da der administrative Aufwand mehrere Betreuungspersonen von ihrer eigentlichen Aufgabe abhält: Der Betreuung der Kinder.



(a) Kinder verschieben ihre Magnete an der Magnettafel.

(b) Betreuungspersonen checken die Kinder vor der Mensa ein.

Abb. 1: Impressionen aus der Schule Freilager

1.2 Fragestellungen & Aufbau der Arbeit

Es stellte sich in der Schule Freilager die Frage, ob eine digitale Lösung diesen administrativen Aufwand übernehmen könnte. Diese Arbeit beschäftigt sich mit der Entwicklung einer solchen Lösung, wobei folgende Fragestellungen beantwortet werden (vgl. Kap. 4.2):

- Ist eine digitale Lösung zur Verwaltung der Betreuung überhaupt notwendig?
- Wie können Betreuungspersonen durch eine digitale Lösung effektiv entlastet werden?
- Wie wird eine digitale Lösung von den Kindern aufgenommen und genutzt?
- Wie sollte das System gestaltet werden, um intuitiv und benutzerfreundlich zu sein?
- Welche Infrastruktur ist für die Umsetzung des Systems erforderlich?
- Welche Programmiersprachen eignen sich für die Entwicklung des Systems?
- Wie kann das System datenschutzkonform umgesetzt werden?
- Wie kann das System flexibel gestaltet werden, um zukünftigen Anforderungen der Schule gerecht zu werden?
- Ist die Investition in das System aus wirtschaftlicher Sicht gerechtfertigt?

Die Arbeit ist in vier thematisch aufeinander aufbauende Teile gegliedert. Im ersten Teil (vgl. Kap. 1) wird nach der dargestellten Ausgangslage und den Fragestellungen im Weiteren eine Bedarfsanalyse folgen, in der untersucht wird, ob der hohe administrative Aufwand auch an anderen Schulen besteht und welche Anforderungen an eine Lösung gestellt werden. Der zweite Teil (vgl. Kap. 2) widmet sich der Methodik, wobei die Entwicklung der digitalen Lösung im Fokus steht. Dabei werden Aspekte wie Hardware, Software, Design, Entwicklung und Datenschutz detailliert erläutert. Im dritten Teil (vgl. Kap. 3) werden das entwickelte System, dessen Funktionen und der Praxiseinsatz beschrieben. Abschliessend werden im vierten Teil (vgl. Kap. 4) die Fragestellungen reflektiert, die Ergebnisse diskutiert und mögliche Weiterentwicklungen des Systems aufgezeigt.

1.3 Bedarfsanalyse: Interviews an Zürcher Tagesschulen

Bevor mit der Entwicklung einer Lösung begonnen werden konnte, war es im Rahmen einer Bedarfsanalyse erforderlich zu klären, ob die Problematik des hohen administrativen Aufwands auch an anderen Schulen auftritt und ob bereits geeignete Lösungen existieren. Da eine Webrecherche keine relevanten Ergebnisse hervorbrachte, wurden an sechs ausgewählten Zürcher Tagesschulen Interviews mit Schulleitungen und Betreuungspersonen durchgeführt. Sinngemässe Transkriptionen der relevanten Fragen und Antworten der Interviews befinden sich im Anhang. Folgende Erkenntnisse wurden gewonnen:

1.3.1 Schule Altstetterstrasse

Das Gespräch mit Frau Hofstetter, Leiterin der Betreuung an der Schule Altstetterstrasse (vgl. Anhang 1), zeigte: Die Magnettafel ist keine Einzellösung. Es wird ein Whiteboard eingesetzt, auf dem die Kinder mit einem Magneten anzeigen können, wohin sie gehen möchten. Eine Betreuungsperson steht den Kindern dabei unterstützend zur Seite (vgl. Abb. 2a). Zusätzlich wird jeden Mittag die Anwesenheitsliste ausgedruckt und aufgehängt, und die Kinder haken sich selbst ab, wenn sie die Mensa betreten. Frau Hofstetter räumte ein: «Wir hätten die Liste gerne digital gemacht, aber das war zu komplex. Wir hätten das gerne umgesetzt, drucken sie aber aktuell aus. Wir haben uns auch schon überlegt, ob wir in Zukunft vielleicht gar nicht mehr kontrollieren sollten. Das wäre jedoch schwierig, weil manchmal ein Anruf von einer Mutter kommt, die fragt: Wo ist mein Kind?»

1.3.2 Schule Allmend

Die Schule Allmend verwendet ebenfalls eine Magnettafel mit Angeboten wie Ruhezimmer, Bibliothek, Chill-Raum, Konstruktion oder Dart (vgl. Abb. 2b). Zur Aufenthaltskontrolle erklärte Franziska Lehnherr, Leiterin der Betreuung (vgl. Anhang 2): «In der Mensa gibt es einen separaten Check-in, bei dem sich die Kinder anstellen und anmelden müssen. Dort haben wir eine ausgedruckte Liste mit allen Kindern, die kommen, und sie werden abgehakt. [...] Das übernimmt jeweils eine Betreuungsperson, die das während der gesamten Mittagszeit macht.» Auf die Frage nach einer digitalen Lösung zeigte sich Frau Franziska Lehnherr grundsätzlich offen, äusserte jedoch Bedenken: «Wir haben überlegt, den Kindern ein Armband zu geben, mit dem sie sich einchecken könnten. Aber das ist datenschutzrechtlich heikel, und wir hatten bisher keine Zeit, diese Idee weiterzuverfolgen.»

1.3.3 Pionier der Magnettafel: Schule Am Wasser

Barbara Stocker, Leiterin der Betreuung an der Schule Am Wasser (vgl. Anhang 3) ist die Erfinderin des Magnettafelsystems. Im Unterstufenhort dieser Schule wird ihre Magnettafel (vgl. Abb. 2c) seit neun Jahren genutzt. Eine Betreuungsperson platziert die Magnete der angemeldeten Kinder auf die Tafel. Die Kinder verschieben ihre Magnete zu den verschiedenen Angeboten, bis sie zum Essen gehen. Beim Essen bringen sie die Magnete in die Küche und legen sie in eine Schüssel, sodass nachvollzogen werden kann, welche Kinder noch nicht gegessen haben und wo sie sich aufhalten. Hier muss täglich eine Betreuungsperson die Magnete anhand der Anwesenheitsliste einzeln vorbereiten. Frau Stocker beschrieb, wie das Magnettafelsystem entstand: «Früher mussten die Kinder sich die Hände waschen und warten, bis alle am Tisch sassen. Wer keine Geduld hatte, machte Unfug, und es gab viel Stress. [...] Dann haben wir das Konzept geändert, sodass die Kinder nicht mehr warten müssen und selbst verantwortlich sind, indem sie ihre Magnete verschieben.»

(a) Magnettafel der Schule Altstetterstrasse

(b) Magnettafel der Schule Allmend

(c) Magnettafel der Schule Am Wasser

Abb. 2: Magnettafeln an ausgewählten Zürcher Tagesschulen

1.3.4 Schule Dachslern-Feldblumen

Die Schule Dachslern-Feldblumen benutzt hingegen keine Magnettafel. Da die Schule keine offenen Betreuungsangebote im klassischen Sinn, sondern Kurse anbietet, lernen die Betreuungspersonen die Kinder mit der Zeit kennen. Die Absenzen werden über eine Drittanbieterlösung von den Eltern erfasst. Schulleiter Karl-Heinz Viethen der Schule Dachslern-Feldblumen (vgl. Anhang 4) berichtete, dass sie gar keine Aufenthaltskontrolle betreiben: «Je mehr wir den Kindern zutrauen, je freier sie sich bewegen können und je weniger sie Magnettäfelchen aufstellen müssen, desto weniger Probleme haben wir», sagte Herr Viethen.

1.3.5 Schule Hutten

Auch die Schule Hutten kommt ohne Magnettafel aus. Carolin Mlakar, Leiterin der Betreuung (vgl. Anhang 5), berichtete: «Wir haben etwa 150 bis 160 Kinder; das ist für Zürcher Verhältnisse noch klein. [...] Eine Magnettafel ist aufgrund der überschaubaren Kinderzahl nicht nötig. Trotzdem wird die Anwesenheitsliste täglich ausgedruckt und abgehakt.»

1.3.6 Eine erste digitale Lösung: QR-Codes an der Schule Lachenzelg

Die Schule Lachenzelg hat eine Vorreiterrolle übernommen und als erste Schule eine digitale Lösung entwickelt. Als Sekundarschule betreut die Schule Kinder im Alter von 12 bis 16 Jahren, d.h. etwas ältere Kinder als die bisher befragten Schulen. Sie verwendet eine selbst entwickelte digitale Lösung basierend auf einer Excel-Liste und einem Barcode-Scanner, die wiederum auf der exportierten Anwesenheitsliste basiert, um die angemeldeten Kinder zu erfassen. An beiden Eingängen zur Mensa wird das System von einer Betreuungsperson bedient. Die Kinder erhalten bei Eintritt in die Schule einen QR-Code bzw. eine Nummer, von denen sie ein Foto auf ihrem Handy speichern. Jeden Mittag werden die QR-Codes von den Betreuungspersonen gescannt. Die Excel-Liste berechnet automatisch, wer noch nicht zum Essen gekommen ist (vgl. Abb. 3). Diese Lösung wurde von Schulleiter Aimen Mokdad (vgl. Anhang 7), ehemals in der IT-Branche tätig, entwickelt. Er erläutert: «Unsere Lösung ist rein aus Bedarf entstanden. Früher druckten wir die Wochenliste, bzw. die Monatsliste aus und die Hortleitung suchte mit dem Bleistift den Namen des Kindes in der Liste und machte einen Strich. Stell dir vor, du machst das mit 380 Kindern. [...] Wir brauchten ein schnelleres System, und so ist diese Liste entstanden.»

Abb. 3: QR-Code-Scanner und Laptop mit Excel-Liste an der Schule Lachenzelg

Daniela Steidle, Leiterin der Betreuung (vgl. Anhang 6), ergänzte: «Seit wir diese Liste vor zwei Jahren eingeführt haben, zeigen viele Schulen grosses Interesse, weil sie eine enorme Erleichterung darstellt. Das Problem ist nur, dass, wenn die Formeln in der Excel-Tabelle nicht korrekt sind, die Berechnungen falsch ausfallen. Wir sind Sozialpädagogen, keine Informatiker, und können solche Fehler nur schwer beheben.»

1.3.7 Zwischenfazit

Die Interviews an den Zürcher Tagesschulen zeigen, dass ein Überblick über den Aufenthaltsort der Kinder als unerlässlich angesehen wird, jedoch mit hohem Aufwand verbunden ist. Schulen wie Altstetterstrasse und Allmend setzen weiterhin auf manuelle Prozesse; Überlegungen zu digitalen Lösungen scheitern an Komplexität oder Datenschutzbedenken. Das Magnettafelsystem der Schule Am Wasser erleichtert die Organisation, bleibt aber zeitaufwendig. Die Schule Dachslern-Feldblumen verzichtet aufgrund ihres Kurssystems auf ein Kontrollsystem, während die Schule Hutten wegen der überschaubaren Kinderzahl keine Magnettafel benötigt. Eine Ausnahme bildet die Schule Lachenzelg, die mit ihrer QR-Code-Lösung erste Schritte in Richtung Digitalisierung geht, allerdings mit technischen Schwierigkeiten konfrontiert ist.
Insgesamt wurde deutlich, dass die bestehenden Systeme, ob analog oder digital, einen erheblichen personellen und administrativen Aufwand erfordern und damit für die Schule Freilager keine vollständige Lösung bereithalten. Ein zuverlässiges, einfaches und datenschutzkonformes digitales System musste neu entwickelt werden.

2 Methodik

Nach der Bedarfsanalyse stand fest: Was bisher die Magnettafel und das Abhaken der Anwesenheitsliste übernommen haben, soll digital ersetzt werden. Die Betreuungspersonen benötigen einen Überblick über die Kinder, um im Zweifelsfall ihren Standort einzusehen. Zudem muss die Anwesenheitsliste in das System integriert werden, um auszuwerten, welche Kinder zur Mittagsbetreuung angemeldet sind und welche die Mensa noch nicht besucht haben. Es wurde in organisatorischer Hinsicht vereinbart, dass die Schule Freilager die technische Infrastruktur zur Verfügung stellt, während der Verfasser dieser Arbeit die Entwicklung des Systems in enger Absprache mit der Schule übernimmt.
Im Folgenden wird erläutert, wie die Entscheidungen hinsichtlich Hardware, Software, Design, Entwicklung und Datenschutz getroffen und umgesetzt wurden. Weiter werden mögliche Kritikpunkte bezüglich der Methodik adressiert.

2.1 Hardware

Das Wichtigste in Kürze:

- Kinder checken sich an jedem Angebot ein und aus, indem sie ihre Hausausweise mit QR-Codes an iPads scannen.
- Die iPads erfassen die Codes mit ihren Kameras, zeigen Informationen über aktuelle Angebote an und dienen den Betreuungspersonen zur Verwaltung.
- Ein separates WLAN verbindet die Geräte, unabhängig vom Schulnetzwerk.
- QR-Codes enthalten nur ID-Nummern, wodurch sensible Daten geschützt bleiben.

Eine Internetrecherche zeigte eine Vielfalt digitaler Tracking-Technologien. Die Möglichkeiten reichen von GPS-Uhren über Gesichtserkennung bis hin zu Biochip-Implantaten. Abgesehen davon, dass diese Lösungen eine unnötige Komplexität mit sich bringen würden, stehen sie bereits aus Datenschutz-Perspektive völlig ausser Frage. Da eine Tagesschule ein öffentliches Organ ist, gelten im Kanton Zürich das *Gesetz über die Information und den Datenschutz (IDG)* [2] und die *Verordnung über die Information und den Datenschutz (IDV)* [3]. Diese Rechtsgrundlagen fordern, dass der Datenschutz bereits bei der Entwicklung von Systemen berücksichtigt wird. Gemäss § 11 IDG steht dabei die Datenminimierung im Vordergrund: Öffentliche Organe müssen Datenbearbeitungssysteme so gestalten, dass nur die für die Aufgabenerfüllung notwendigen Personendaten erhoben werden.
Es wurde eine datenschutzrechtlich konforme Lösung benötigt. Den genauen Standort der Kinder zu tracken, wäre unzweckmässig. Die Betreuungspersonen müssen lediglich wissen, in welchem Angebot sich das Kind aktuell befindet, ähnlich wie bei der Magnettafel. Es braucht daher an jedem Angebot ein Gerät, das die Kinder ein- oder auschecken kann. Um das Erfassen von biometrischen Daten zu vermeiden, sollen die Kinder durch ein physisches Identifikationsmittel unterschieden werden. Unter Berücksichtigung dieser Grundsätze blieben drei sinnvolle Möglichkeiten (vgl. Abb. 4): *Radio-Frequency Identification (RFID)*-Tags [4], *Near Field Communication (NFC)*-Tags [5] und Barcodes [6] bzw. *Quick Response Codes (QR-Codes)* [7]. Alle drei Technologien haben den Vorteil, einen kurzen Text analog repräsentieren zu können und dafür keine Energiequelle zu benötigen. Auf dem Identifikationsmittel würde lediglich eine ID-Nummer abgebildet, die auf dem Server einem Kind zugeordnet ist. Beim Auslesen durch eine unbefugte Person werden somit keine sensiblen Daten preisgegeben.

2.1.1 Die Identifikationsmittel

RFID-Tags werden oft im Einzelhandel eingesetzt, da sie von Lesegeräten aus mehreren Metern Entfernung identifiziert werden können. Ein ähnliches System könnte in der Schule funktionieren: Wenn die Kinder ein Angebot betreten oder verlassen, können sie passiv vom Server ein- und ausgecheckt werden. Während die Tags für wenige Franken erhältlich sind [8], kosten gewisse Lesegeräte indessen über tausend Franken pro Stück [9]. Für ca. 10 Angebote in der Schule Freilager wäre dies eine erhebliche Investition. Zudem muss jedes Angebot mit einem Lesegerät ausgestattet werden, was das Hinzufügen spontaner Angebote unmöglich macht.

NFC-Tags funktionieren ähnlich wie RFID-Tags, können jedoch nur auf kurze Distanzen im Zentimeterbereich gelesen werden. Sie sind häufig in Kreditkarten verbaut. Die entsprechenden Lesegeräte sind deutlich günstiger; viele Smartphones und Tablets können NFC-Tags lesen. Die Kinder könnten einen solchen NFC-Tag in einem Armband tragen und sich an den einzelnen Angeboten aktiv ein- und auschecken, indem sie das Armband an ein vor Ort aufgestelltes Gerät halten.

Barcodes bzw. QR-Codes kodieren Text in ein visuelles Format, das von jeder Kamera mit entsprechender Dekodierungssoftware gelesen werden kann. Die Kinder müssten sich, wie bei den NFC-Tags, an den Angeboten ein- und auschecken, indem sie den Code von einem dort aufgestellten Gerät scannen lassen. Dieses System wurde aus folgenden Gründen für die Schule Freilager gewählt: Barcodes und QR-Codes sind die mit Abstand kostengünstigste und flexibelste Lösung, es werden keine unnötigen Daten gesammelt, es erfordert keine komplexe Technologie, und die Codes können schnell erstellt sowie ersetzt werden. Jedes Kind in einer Zürcher Volksschule besitzt bereits eine siebenstellige User-ID-Nummer, die von *Kommunikations- und Informations-Technologien für die Schulen der Stadt Zürich (KITS)* [10] generiert wird. Diese ID sieht etwa so aus: `A123456`. Sie dient nicht nur als Login für KITS-Geräte, sondern wird auch von der Schule für administrative Zwecke verwendet, beispielsweise für das Ausleihen von Büchern in der Bibliothek. Um keine neuen unnötigen ID-Nummern zu generieren und Duplikate zu vermeiden, wurde auf diese User-IDs zurückgegriffen. Diese wurden mit einem Kartendrucker auf Plastikkarten gedruckt, die die Kinder während der Mittagszeit als Hausausweis an Schlüsselbändern um den Hals tragen (vgl. Kap. 3.7). Sie werden ausserhalb der Nutzungszeiten im Klassenzimmer deponiert. Ein Kartendrucker ist zwar anfangs eine Investition, ermöglicht aber das Drucken neuer Karten zu minimalen Kosten. Da ursprünglich angenommen wurde, dass der Scanner in der Bibliothek nur Barcodes lesen kann, wurden zunächst Barcodes anstelle von QR-Codes auf die Hausausweise gedruckt. Später wurde jedoch auf QR-Codes umgestellt (vgl. Kap. 3.4).

RFID-Tags	**NFC-Tags**	**Barcodes / QR-Codes**
• Lesbar aus mehreren Metern Entfernung. • Passives Ein-/ Auschecken möglich. • Kostenintensive Lesegeräte. • Ersetzen oder Hinzufügen von Tags erfordert Einprogrammierung.	• Lesbar aus wenigen Zentimetern. • Aktives Ein-/ Auschecken nötig. • Günstige Lesegeräte, oft integriert. • Ersetzen oder Hinzufügen von Tags erfordert Einprogrammierung.	• Lesbar aus wenigen Zentimetern. • Aktives Ein-/ Auschecken nötig. • Kamera genügt als Lesegerät. • Einfaches Ersetzen oder Hinzufügen ohne Einprogrammierung.

Abb. 4: Vergleich zwischen möglichen Identifikationsmitteln

2.1.2 Die Lesegeräte

Weiter stellte sich die Frage, welche Geräte an den Angeboten aufgestellt werden sollten. Da die Betreuungspersonen unterstützt und nicht behindert werden müssen, wurden manuelle Lösungen, wie ein Barcode-Scanner, ausgeschlossen. Ein Gerät mit ständig laufender Kamera und Touch-Bildschirm wäre optimal: Die Kinder könnten sich ein- und auschecken, sowie Informationen über das aktuelle Angebot und weitere laufende Angebote mit freien Plätzen sehen. Ausserdem sollten die Betreuungspersonen diese Geräte gleichzeitig als Dashboard nutzen können, um den Standort der Kinder einzusehen, Angebote zu verwalten und die Anwesenheitsliste einzupflegen. Die Wahl fiel auf iPads (vgl. Kap. 3.7), denn sie sind benutzerfreundlich, schnell aufgestellt und abgebaut, haben eine integrierte Kamera und bieten genug Platz, um Informationen über weitere Angebote und Personen anzuzeigen. Die iPads werden kurz vor Mittag von den Betreuungspersonen geholt und am Ort der Angebote, z.B. am Eingang der Turnhalle, aufgestellt. Nach Gebrauch werden sie im Teamzimmer deponiert und aufgeladen.

2.1.3 Das Netzwerk

Die Hausausweise und iPads allein bilden noch kein vollständiges System. Um Informationen über andere Angebote anzuzeigen, sollten die iPads miteinander kabellos kommunizieren. Übliche Technologien dafür sind *Bluetooth* [11], mobile Daten [12] und *Wireless Local-Area Network (WLAN)* [13]. Bluetooth hat jedoch nur eine Reichweite von wenigen Metern, was bei weit auseinanderliegenden Angeboten nicht funktioniert. Mobile Daten wurden ebenfalls ausgeschlossen, denn im Grossteil des Schulgebäudes gibt es kein Netz. Somit blieb WLAN in der Schule Freilager als einzige und wohl auch beste Option. Während die Schule bereits über WLAN verfügt, eingerichtet von der *Organisation und Informatik Zürich (OIZ)* [14], ist dieses mit Firewalls gesichert und würde zur Verbindung von Geräten für ein solches System eine Genehmigung erfordern. Um diesen bürokratischen Aufwand vorerst zu minimieren und kein Risiko einzugehen, durch versehentliche Konfigurationsänderungen den Betrieb des bestehenden Netzwerkes zu stören, wurde ein zweites, lokales Netzwerk im Schulhaus aufgebaut. Die Schule Freilager stellte dafür einen handelsüblichen TP-Link-Router mit mehreren Mesh-Verstärkern auf (vgl. Kap. 3.7), um alle nötigen Bereiche abzudecken. Ein WLAN-Netzwerk funktioniert auch ohne Internet. Es können dabei nur mit dem Netzwerk verbundene Geräte miteinander kommunizieren. Dies spart monatliche Internetanbieter-Kosten und verbessert den Datenschutz, da sich nur Geräte im lokalen Netzwerk untereinander austauschen können. Es wurde damit eine sogenannte On-Premises-Lösung entwickelt: Die Daten werden nicht in der Cloud, sondern lokal in der eigenen Infrastruktur gespeichert und verarbeitet (vgl. Abb. 5). Dies gibt mehr Kontrolle über die Daten, erfordert aber auch, dass die gesamte Hardware und Software vor Ort verwaltet wird.

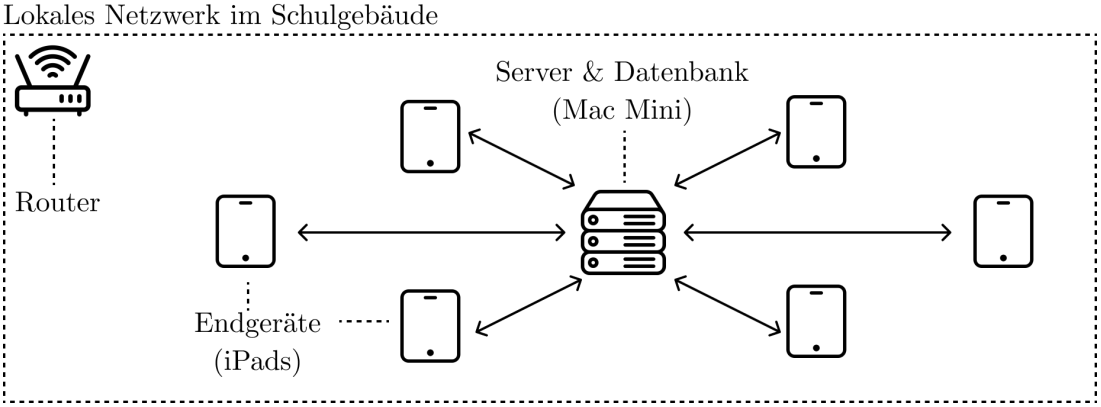


Abb. 5: Struktur der On-Premises-Lösung

2.1.4 Der Server & die Datenbank

Um die Daten lokal zu speichern und zentral zu verwalten (vgl. Kap. 2.5), wurde ein Server eingesetzt. Dieser übernimmt die Aufgabe, alle Anfragen der iPads zu bearbeiten und die Daten in einer Datenbank zu speichern. Ein Server dient als zentrale «Source of Truth» – alle Daten werden an einem Ort gesammelt und können jederzeit von den iPads abgerufen werden. Als Server wurde ein Mac Mini ausgewählt, der rund um die Uhr läuft und keinen Bildschirm benötigt. Der Mac Mini bietet genug Leistung für diese Aufgaben, ist kompakt und energieeffizient (vgl. Kap. 3.7).

Um die Daten auf dem Server effizient zu speichern und zu verwalten, wird eine Datenbank benötigt. Eine Datenbank ist eine strukturierte Sammlung von Informationen, die es ermöglicht, Daten schnell zu speichern, zu organisieren und bei Bedarf abzurufen. Sie liegt auf dem Server, dem Mac Mini, und dient als «Gedächtnis» der gesamten Lösung.
Während der Server dafür zuständig ist, Anfragen von den iPads zu empfangen und diese zu bearbeiten, ist die Datenbank der Ort, an dem die tatsächlichen Daten, z.B. Informationen zu den Angeboten und den Check-ins der Kinder, gespeichert werden. Der Server kommuniziert also mit der Datenbank, um die benötigten Informationen zu speichern oder abzurufen.
Für die Lösung an der Schule Freilager wurde SQLite als Datenbank genutzt. SQLite ist ressourcenschonend, schnell und ideal für Anwendungen, die nicht mit riesigen Datenmengen arbeiten müssen [15]. Da es direkt auf dem Mac Mini läuft, benötigt es keine separate Server-Infrastruktur, was die Verwaltung vereinfacht. Diese Kombination aus Server und Datenbank ermöglicht es, die Daten zügig zu verarbeiten und sicher lokal zu speichern (vgl. Abb. 6).

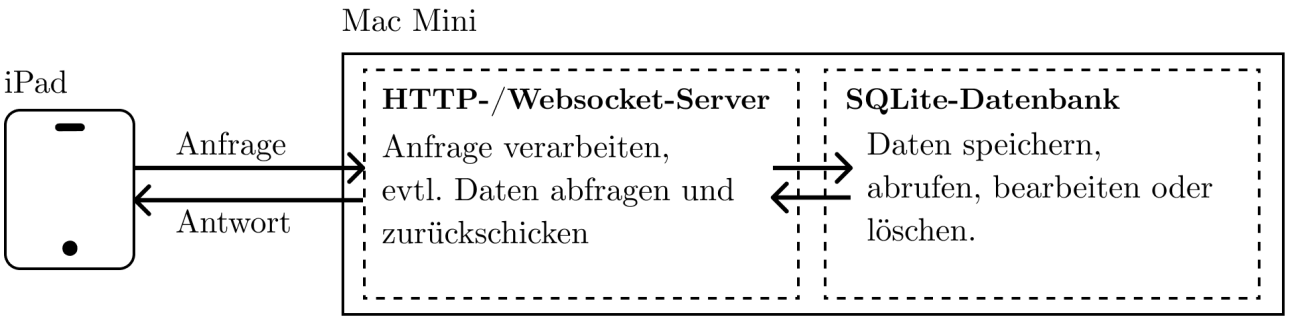


Abb. 6: Schematische Darstellung des Servers mit Datenbank

2.2 Software

Das Wichtigste in Kürze:

- Die App ist in zwei Ebenen unterteilt: Kinder-Ebene für das Ein- und Auschecken sowie das Anzeigen von Angeboten und Dashboard-Ebene für die Verwaltung durch Betreuungspersonen.
- Die App nutzt SwiftUI, ein modernes deklaratives Framework, und folgt dem MVVM-Architekturprinzip zur klaren Trennung von Logik und Benutzeroberfläche.
- Der Server läuft auf einem Mac Mini und wurde mit Swift und dem Vapor-Framework entwickelt, um eine effiziente Kommunikation mit der App zu ermöglichen.
- Eine SQLite-Datenbank wird über das ORM Fluent verwaltet, um Datenbankabfragen effizient zu strukturieren.
- WebSockets ermöglichen Echtzeit-Updates, z.B. die Anzeige der freien Plätze in Angeboten, ohne unnötigen Netzwerkverkehr zu erzeugen.
- Dank einer geteilten Codebasis können Modelle und Schnittstellen zwischen Server und App wiederverwendet werden.

2.2.1 Die App

Auf der Grundlage der definierten technischen Voraussetzungen wurde eine Applikation für die Schule Freilager (im Folgenden «die App» genannt) konzipiert, die auf den iPads laufen soll. Eine App ist ein Softwareprogramm, das spezifische Aufgaben auf einem Gerät, wie einem Smartphone oder Tablet, ausführt. In diesem Fall wird sie in die Kinder-Ebene und die Dashboard-Ebene unterteilt.
Auf der Kinder-Ebene werden das aktuelle Angebot, die freie Platzanzahl von anderen Angeboten und die Selfie-Kamera angezeigt. Kinder scannen beim Betreten und Verlassen eines Angebots ihren Hausausweis mit der integrierten Selfie-Kamera des iPads, um sich ein- oder auszuchecken.
Betreuungspersonen und Systemadministratoren erhalten hingegen beim Scannen ihres Ausweises Zugriff auf die Dashboard-Ebene. Bei Administratoren ist zusätzlich die Eingabe eines Codes erforderlich. Das Dashboard ermöglicht ihnen, Angebote zu übernehmen, die maximale Teilnehmerzahl anzupassen, nach Personen zu suchen und anzuzeigen, wer in der Mensa noch nicht gegessen hat. Systemadministratoren haben zusätzlich die Möglichkeit, die Anwesenheitsliste hochzuladen, Angebote und Personen hinzuzufügen und zu bearbeiten sowie weitere administrative Optionen zu nutzen (vgl. Abb. 7).

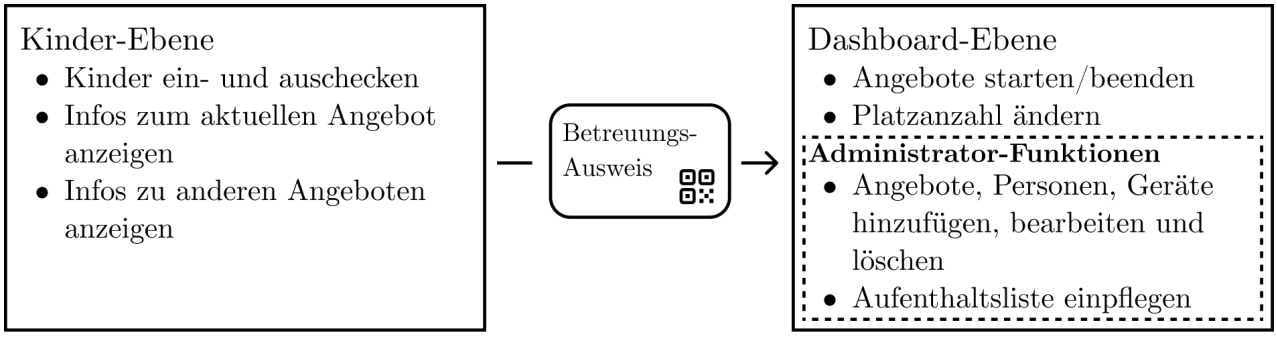


Abb. 7: Struktur der App mit Kinder- und Betreuungsebene

2.2.1.1 SwiftUI

Seit ihrer Vorstellung auf der Worldwide Developers Conference 2019 [16] wird Apples hauseigene Programmiersprache SwiftUI eine immer beliebtere Lösung zur Entwicklung von Apps auf Apple-Geräten. SwiftUI ist ein deklaratives UI-Framework, bei dem der Entwickler beschreibt, wie die Oberfläche aussehen und was sie tun soll, anstatt die Schritte zur Erstellung dieser Oberfläche detailliert zu definieren; das System kümmert sich um den Rest [17]. SwiftUI basiert auf Swift [18], einer robusten und intuitiven Programmiersprache, entwickelt von Apple. Swift ist Open Source, d.h., dass Entwickler weltweit den Swift-Quellcode ansehen, modifizieren und nutzen können. Im Gegensatz dazu ist SwiftUI nicht Open Source, was bedeutet, dass der Quellcode von SwiftUI nicht öffentlich zugänglich ist und nur von Apple weiterentwickelt und gepflegt werden kann. Apps mit SwiftUI werden in Apples eigener Entwicklungsumgebung Xcode [19] erstellt und programmiert.
Apps können auch mit anderen Programmiersprachen wie Kotlin [20] oder JavaScript mit React Native [21] entwickelt werden. Kotlin wird oft für Android-Apps genutzt, da es speziell auf diese Plattform abgestimmt ist. JavaScript in Kombination mit Frameworks wie React Native erlaubt die Entwicklung plattformübergreifender Apps, die sowohl auf iOS als auch auf Android laufen. Diese Ansätze eignen sich, wenn Apps auf mehreren Geräten laufen sollen, bringen jedoch oft Kompromisse in der Anpassung oder Leistung mit sich. Für dieses Projekt wurde SwiftUI gewählt, da die App ausschliesslich auf iPads laufen soll.

2.2.1.2 Architektur

Apple bietet zwar viele Tutorials zu den verschiedenen Funktionen von SwiftUI an, jedoch fehlen Informationen oder Empfehlungen zur Code-Architektur, d.h. zur Strukturierung des Codes. Es gibt keine universelle Architektur; sie variiert von Projekt zu Projekt. Vorliegend wurde eine vom Verfasser angepasste Version des *Model–View–ViewModel (MVVM)*-Patterns verwendet [22]. Die Grundstrukturen sind dabei in `Views`, `Models` und `ViewModel` unterteilt, welche alle unterschiedliche Funktionen übernehmen (vgl. Abb. 8).

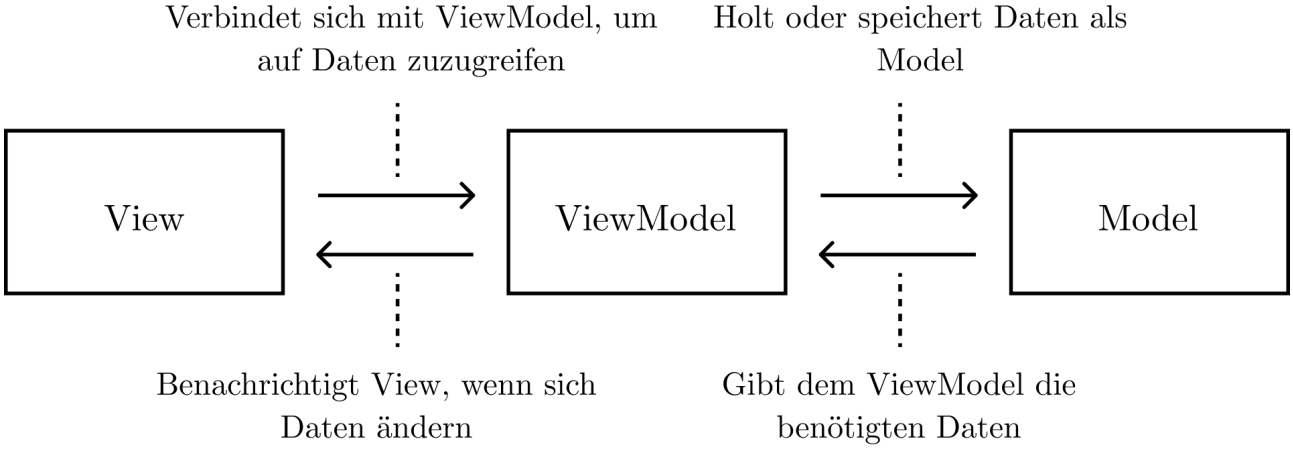


Abb. 8: Aufbau der MVVM-Architektur: Model, View und ViewModel

2.2.1.3 Views, View Containers und Modifiers

Apple beschreibt Views als die grundlegenden Bausteine von Benutzeroberflächen. In einer App wird alles, was angezeigt wird, durch eine View definiert [23]. Views sind Computercode, geschrieben mit SwiftUI, der beschreibt, wie etwas grafisch aussehen und sich verhalten soll. SwiftUI bietet eine Vielzahl von sogenannten primitiven, vordefinierten Views, wie `Color`, `Text`, `RoundedRectangle` und `Image`. Diese Views können mit Parametern angepasst werden, z.B. der

angezeigte Text, die Aktion eines Knopfes oder der Name eines angezeigten Bildes. Primitive Views sind die einfachsten Bausteine in SwiftUI. Sie repräsentieren grundlegende Elemente wie Text, Bilder, Formen und Farbe. Das folgende Beispiel verdeutlicht die Nutzung von `Text`, `Color` und `RoundedRectangle`:

Code:

```
Color.green
Text("3 Freie Plätze")
RoundedRectangle(cornerRadius: 12)
```

Resultat:

View Containers sind spezielle Views, die andere Views enthalten können. Sie helfen dabei, das Layout der Benutzeroberfläche zu organisieren. Zum Beispiel können `VStack`, `HStack` und `ZStack` Views untereinander, nebeneinander und übereinander anordnen. Zusätzlich helfen `List`, `GroupBox`, `Label` und mehr, die beinhalteten Views grafisch anzupassen. Eine `List` kann auf iOS, dem Betriebssystem für mobile Geräte von Apple [24], und macOS, dem Betriebssystem für Apple-Computer [25], unterschiedlich aussehen, obwohl sie mit dem gleichen Code definiert wurde. Beispielsweise lassen sich mit einem `HStack` ein Text und ein Bild horizontal anordnen:

Code:

```
HStack {
    Text("3 Freie Plätze")
    Image(systemName: "plusminus")
}
```

Resultat:

Um Views graphisch anzupassen, können Modifiers benutzt werden. Beispielsweise können die Schriftart geändert, ein Rand hinzugefügt oder Ecken abgerundet werden. Ein Modifier ist eine Methode, die eine neue, angepasste View aus einer bestehenden View erstellt. Dabei spielt die Reihenfolge der Modifiers eine Rolle. Wenn `.padding(16)`, d.h. ein Rand mit Grösse 16, vor dem Hintergrund `.background(Color.green)` hinzugefügt wird, ist der Hintergrund sichtbar grösser. Falls diese Reihenfolge umgekehrt ist, wird der Rand ausserhalb des Hintergrunds hinzugefügt:

Code:

```
HStack {
    Text("Freie Plätze: 3")
    Image(systemName: "plusminus")
}
.padding(16)
.background(
    Color.green
)
```

Resultat:

Code:

```
HStack {
    Text("Freie Plätze: 3")
    Image(systemName: "plusminus")
}
.background(
    Color.green
)
.padding(16)
```

Resultat:

Neben diesen vordefinierten Views lassen sich Views selbst erstellen, aus weiteren Views, View Containers und Modifiers. Diese können daraufhin, genau wie `Text`, `Color` oder `VStack` beliebig oft verwendet werden. Dabei können sie ebenfalls Parameter annehmen, d.h. Variablen, die beispielsweise den angezeigten Text verändern können. Im folgenden Beispiel wird eine `FreiePlätzeView` erstellt, die eine Variable `anzahlPlätze` als Parameter erwartet und diese neben einem Text «Freie Plätze:» anzeigt:

Code:

```
struct FreiePlätzeView: View {
    var anzahlPlätze: Int
    var body: some View {
        HStack {
            Text("Freie Plätze: ")
            Text(anzahlPlätze, format: .number)
            Image(systemName: "plusminus")
        }
        .padding(16)
        .background(
            Color.green
        )
    }
}

// Nutzung:
FreiePlätzeView(anzahlPlätze: 12)
```

Resultat:

Dazu können Views auch `@State` verwenden, um den Zustand einer Variable zu verwalten und die Benutzeroberfläche dynamisch zu aktualisieren. Dies ermöglicht, interaktive Elemente zu erstellen, die auf Benutzereingaben reagieren. Im folgenden Beispiel wird die mit `@State` markierte Variable `anzahlPlätze` beim Drücken auf einen `Button`, also einen Knopf, erhöht:

Code:

```
struct FreiePlätzeView: View {
    @State var anzahlPlätze: Int // Der aktuelle Zustand, wie viele Plätze frei sind

    var body: some View {
        VStack(alignment: .leading) {
            HStack {
                Text("Freie Plätze: ")
                Text(anzahlPlätze, format: .number) // Zeigt die Anzahl der freien Plätze an
                    .fontWeight(.bold)
            }

            Button(action: {
                anzahlPlätze += 1 // Erhöht die Anzahl der freien Plätze um 1
            }) {
                HStack {
                    Image(systemName: "plus")
                    Text("Platz hinzufügen")
                }
            }
            .buttonStyle(.bordered)
        }
    }
}

// Nutzung:
FreiePlätzeView(anzahlPlätze: 12)
```

Resultat:

Freie Plätze: **12**

+ Platz hinzufügen

Resultat nach Knopfdruck:

Freie Plätze: **13**

+ Platz hinzufügen

Die Benutzeroberfläche der App besteht schlussendlich aus vielen verschachtelten Views. Das Ziel dabei ist, möglichst wenig repetitiven Code zu schreiben. Um aber beispielsweise eine wiederverwendbare `AngebotView` zu ermöglichen, müssen die Informationen über das anzuzeigende Angebot an einem bestimmten Ort gespeichert sein – etwa Titel, Beschreibung, freie Plätze etc. Diese Daten werden in sogenannten Models verwaltet.

2.2.1.4 Models

In Swift können Models als einfache Strukturen (`struct`) oder Klassen (`class`) definiert werden. Sie sind unabhängig von der Benutzeroberfläche und fokussieren sich rein auf die Daten und deren Manipulation. Das folgende Model beschreibt die Struktur eines Angebots mit Titel, Beschreibung und Anzahl freier Plätze:

Code:

```
struct Angebot {
    var titel: String
    var beschreibung: String
    var freiePlätze: Int
}
```

Das Model sorgt dafür, dass diese Informationen an einem zentralen Ort gesammelt sind. Die App greift auf diese Daten zurück, um sie in der Benutzeroberfläche anzuzeigen. Es hilft, die App übersichtlicher zu gestalten, da die Daten klar von der Anzeige getrennt sind. Damit lässt sich eine `AngebotView` erstellen, die ein beliebiges `Angebot` als Parameter akzeptiert:

Code:

```
struct AngebotView: View {
    var angebot: Angebot

    var body: some View {
        VStack(alignment: .leading) {
            Text(angebot.titel)
                .font(.title3.bold())

            Text(angebot.beschreibung)

            Text("Freie Plätze: "(angebot.freiePlätze)")
                .foregroundStyle(.secondary)
        }
    }
}

// Nutzung:
AngebotView(angebot: Angebot(titel: "Mensa", beschreibung: "Hamburger mit Pommes",
freiePlätze: 12))
```

Resultat:

Mensa
Hamburger mit Pommes
Freie Plätze: 12

Allerdings fehlt noch eine Quelle, aus der die Angebote stammen. Die Daten müssen vom Server auf das Gerät geladen werden. Dieser Prozess – das Abfragen der Daten vom Server und das Speichern der empfangenen Informationen – passiert im ViewModel.

2.2.1.5 ViewModel

Das ViewModel ist die Brücke zwischen den Daten (Models) und der Benutzeroberfläche (Views). Es kümmert sich darum, die Daten vom Server abzurufen, zu speichern und dann der View zur Verfügung zu stellen. Es trennt die Logik und Datenverarbeitung von der Darstellung, was den Code übersichtlicher und wiederverwendbarer macht. Das folgende ViewModel, definiert als Klasse, beinhaltet eine Variable `angebote`, in der die geladenen `Angebot`-Models gespeichert werden, und eine Methode `ladeAngebote`, die den Serverabruf ausführt und die geladenen Angebote in der Variable speichert:

Code:

```
class ViewModel: ObservableObject {
    @Published var angebote: [Angebot] = []

    func ladeAngebote() {
        // Serverabruf
        empfangeneAngebote = API().getAngebote()

        // Die empfangenen Daten werden gespeichert
        self.angebote = empfangeneAngebote
    }
}
```

Durch das Zusammenspiel von ViewModel, Model und View können die Daten vom Server abgerufen, im ViewModel gespeichert und in der Benutzeroberfläche angezeigt werden. Das ViewModel lädt die Daten vom Server und speichert diese in einem Model. Die View greift auf die Daten im ViewModel zu und zeigt sie entsprechend an. Dadurch bleibt die Logik der Datenverarbeitung von der Anzeige getrennt, und die Daten können effizient verwaltet und

dynamisch dargestellt werden. Während in der typischen MVVM-Architektur ein ViewModel für jede View erstellt wird [22], wurde vorliegend nur ein globales ViewModel genutzt, da die Datenmenge relativ gering und simpel ist. Das folgende Beispiel verdeutlicht, wie dieses Zusammenspiel tatsächlich implementiert wird. In der View wird das ViewModel als Variable erstellt. Darauf kann in der Liste auf `viewModel.angebote` zugegriffen werden. Die Methode, um diese Angebote zu laden, wird beim Erscheinen der View aufgerufen:

Code:

```
struct AngebotListeView: View {
     // Erstellung des ViewModels, das den Zustand verwaltet:
    @StateObject var viewModel = ViewModel()

    var body: some View {
        VStack {
            // Liste, die durch `viewModel.angebote` iteriert:
            List(viewModel.angebote, id: \.titel) { angebot in
                AngebotView(angebot: angebot)
            }
        }
        .onAppear {
            // Lädt die Angebote-Daten beim Erscheinen der View:
            viewModel.ladeAngebote()
        }
        // Stellt das ViewModel für untergeordnete Views bereit:
        .environmentObject(viewModel)
    }
}
```

2.2.2 Das Backend

Das ViewModel greift auf die Daten des Servers zu, um die notwendigen Informationen in der App darzustellen. Dieser Server, in der Schule Freilager der Mac Mini, muss jedoch so programmiert sein, dass er weiss, wie er Anfragen der iPads verarbeiten soll.

2.2.2.1 Swift on Server & Vapor

Während klassische Serveranwendungen mit etablierten Sprachen wie PHP [26], Python [27] oder Ruby [28] entwickelt werden, wurde sich in diesem Projekt für die Verwendung von Swift als Programmiersprache entschieden. Swift ist vor allem für die Entwicklung von iOS- und macOS-Apps bekannt, bietet jedoch auch die Möglichkeit, auf Servern genutzt zu werden [29]. Swift ist eine schnelle und sichere Sprache mit einer modernen Syntax, die sowohl auf macOS als auch auf Linux läuft. Durch die Verwendung von Swift für das Backend konnte die gleiche Programmiersprache sowohl für die App als auch für den Server verwendet werden.
Um Swift als Server-Sprache zu nutzen, wird ein Framework benötigt, das die Kommunikation mit dem Web ermöglicht, wie zum Beispiel das Beantworten von *Hypertext Transfer Protocol (HTTP)*-Anfragen [30]. Hier kommt Vapor [31] ins Spiel. Vapor ist ein leistungsstarkes Web-Framework, das es ermöglicht, Serveranwendungen in Swift zu schreiben. Es unterstützt Funktionen wie HTTP-Anfragen und WebSockets und bietet eine einfache Integration mit Datenbanken wie SQLite oder PostgreSQL [32]. Dank Vapor konnte auf umfangreichen vorlagenhaften Code verzichtet und direkt mit der Implementierung eines effizienten Servers begonnen werden, der die Daten von den iPads verarbeitet und mit der App kommuniziert. Ein Vapor-Server kann folgendermassen gestartet werden:

Code:

```
import Vapor

// Erstellung einer neue Vapor-Anwendung:
let app = try await Application.make(.detect())

// Startet die Anwendung und verarbeitet eingehende Anfragen:
try await app.execute()
```

2.2.2.2 HTTP & WebSockets

Die Kommunikation zwischen den iPads und dem Server muss nach einem klar definierten Protokoll ablaufen, damit beide Seiten wissen, wie die Informationen verarbeitet werden sollen. Eines der bekanntesten und am häufigsten genutzten Protokolle ist HTTP. HTTP-Anfragen werden verwendet, um Daten zwischen einem Client (wie einem Webbrowser oder der App) und einem Server auszutauschen. Immer wenn eine Website im Browser geladen wird, erfolgt dieser Austausch in der Regel über HTTP.
Eine sicherere Variante davon ist *Hypertext Transfer Protocol Secure (HTTPS)* [33], bei dem die Daten zusätzlich verschlüsselt werden, um die Vertraulichkeit und Integrität der Informationen zu gewährleisten. HTTPS wird vor allem bei sensiblen Daten verwendet, um sicherzustellen, dass Unbefugte keine Einsicht in die übermittelten Informationen erhalten.
Vorliegend wurde jedoch nur HTTP verwendet, da es sich um eine geschlossene, lokale Netzwerkumgebung handelt, in der keine externen Sicherheitsrisiken durch das Internet bestehen. Da alle Anfragen innerhalb eines abgesicherten Netzwerks bleiben, ist eine zusätzliche Verschlüsselung durch HTTPS nicht notwendig. In Vapor kann eine HTTP-Route unkompliziert erstellt werden:

Code:

```
// Registriere eine GET-Route unter dem Pfad "/hello":
app.get("hello") { req in
    // Wenn die Route aufgerufen wird, gib den Text "Hello, world!" als Antwort zurück:
    return "Hello, world!"
}
```

HTTP-Anfragen eignen sich hervorragend, um einzelne Anfragen von der App an den Server zu senden, z.B. wenn ein Angebot übernommen wird. Allerdings benötigt die App ständig aktualisierte Daten, wie etwa die aktuelle Anzahl freier Plätze in einem Angebot. Eine Möglichkeit, dies zu lösen, wäre, dass die App in regelmässigen Abständen (z.B. alle paar Sekunden) eine HTTP-Anfrage an den Server sendet, um die neuesten Daten abzufragen. So war es ursprünglich auch implementiert.
Allerdings stellte sich heraus, dass dieser Ansatz ineffizient ist, da er unnötigen Netzwerkverkehr erzeugt und die Serverlast erhöht (vgl. Kap. 3.2). Eine deutlich effizientere Methode ist der Einsatz von WebSockets [34]. WebSockets ermöglichen eine dauerhafte Verbindung zwischen der App und dem Server, wodurch ein «Daten-Tunnel» entsteht. Durch diesen Tunnel können Daten in Echtzeit übertragen werden, ohne dass ständige neue Anfragen nötig sind (vgl. Abb. 9). Sobald sich also etwas in der Datenbank ändert, wie die Anzahl der freien Plätze, kann der Server die aktualisierten Informationen sofort an die App senden.
Das bedeutet, dass alle Anfragen von der App an den Server, wie etwa das Übernehmen eines Angebots, weiterhin über HTTP-Anfragen gesendet werden. HTTP wird für jede Art von Interaktion genutzt, bei der die App dem Server eine bestimmte Aufgabe mitteilt.
Sobald jedoch Datenaktualisierungen erforderlich sind, wie etwa die Anzahl freier Plätze oder Änderungen in der Datenbank, werden diese Updates vom Server über WebSockets in Echtzeit

an die App gesendet. Dadurch erhält die App die neuesten Informationen sofort und ohne Verzögerung, während die eigentlichen Anfragen und Aktionen nach wie vor über das bewährte HTTP-Protokoll abgewickelt werden. Vapor erleichtert die Implementierung von WebSockets ebenfalls. Folgendermassen kann eine WebSocket-Route erstellt werden:

Code:

```
// WebSocket-Route unter "/WebSocket"
app.webSocket("WebSocket") { req, ws in
    // Verbindung hergestellt
}

// Nachricht "updates" an den Client senden
ws.send("updates")
```

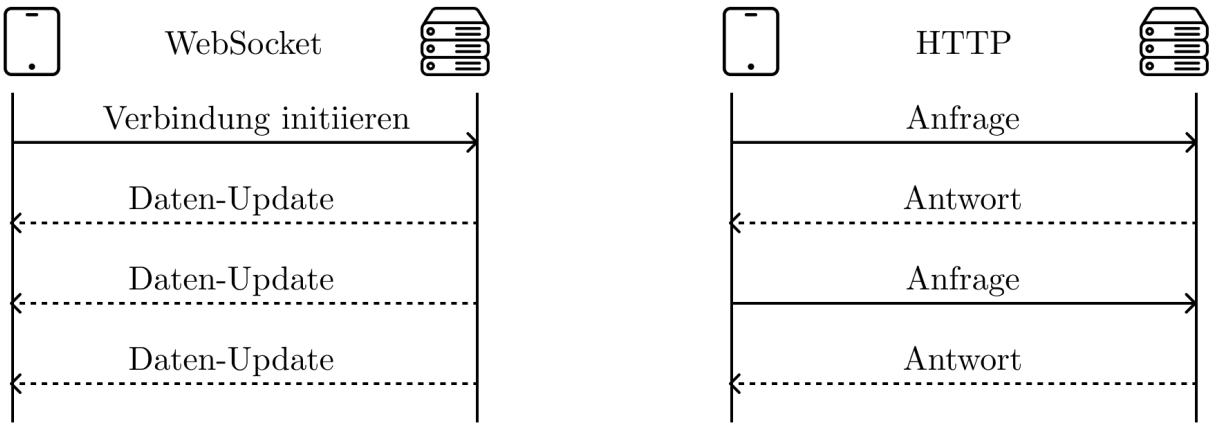


Abb. 9: Vergleich von WebSocket- und HTTP-Anfrage

2.2.2.3 SQLite mit Fluent

Vapor erleichtert die Interaktion mit der Datenbank erheblich. Die Arbeit mit Datenbanken erfordert normalerweise viel Code, um sicherzustellen, dass alle Vorgänge korrekt und effizient ablaufen. Da Vapor jedoch Fluent [35] verwendet, ein *Object-Relational Mapping (ORM)* Framework, wird dieser Prozess jedoch stark vereinfacht. ORM ermöglicht, die Datenbankstrukturen als Objekte in der Programmiersprache abzubilden. Auf diese Weise können Datenbankabfragen in einer für den Entwickler vertrauten Syntax geschrieben werden, ohne dass direkt *Structured Query Language (SQL)* [36], eine standardisierte Sprache zur Verwaltung und Manipulation von Daten in Datenbanken, verwendet werden muss.

Fluent stellt sicher, dass grundlegende Datenbankoperationen wie das Erstellen, Lesen, Aktualisieren und Löschen von Daten auf einfache Weise durchgeführt werden können. Dies ermöglicht die Interaktion mit der Datenbank, ohne sich um die Feinheiten der SQL-Abfragen kümmern zu müssen. Ein weiterer Vorteil von Fluent ist die Flexibilität bei der Wahl der Datenbank. Es erlaubt die Verwendung unterschiedlicher Datenbanktypen, ohne dass grössere Code-Änderungen notwendig sind. Sollte beispielsweise der Wechsel von einer SQLite-Datenbank zu einer MongoDB-Datenbank [37] erforderlich sein, lässt sich dies mit minimalem Aufwand umsetzen.

Der folgende Code-Ausschnitt zeigt ein `AngebotModel` mit verschiedenen Parametern, das sich mit Fluent in einer Datenbank speichern lässt:

Code:

```
import Fluent

// Modell für die Datenbank-Tabelle "angebote":
final class AngebotModel: Model, Content {
    static let schema = "angebote" // Name der Tabelle in der Datenbank.

    @ID(key: .id) // Primärschlüssel.
    var id: UUID?

    @Field(key: "titel") // Spalte "titel" in der Tabelle.
    var titel: String

    @Field(key: "beschreibung") // Spalte "beschreibung" in der Tabelle.
    var beschreibung: String

    @Field(key: "aktiviert") // Spalte "aktiviert", speichert einen Bool-Wert.
    var aktiviert: Bool

    init() {}

    // Initializer, um ein neues Angebot zu erstellen.
    init(id: UUID? = nil, titel: String, beschreibung: String, aktiviert: Bool = true) {
        self.id = id
        self.titel = titel
        self.beschreibung = beschreibung
        self.aktiviert = aktiviert
    }
}

// Speichern eines neuen Angebots in der Datenbank:
let angebot = Angebot(titel: "Ruheraum", beschreibung: "Stille!")
try await angebot.save(on: req.db)
```

Um wiederkehrenden Code zu vermeiden und eine saubere Trennung von Server-Logik und Datenbankoperationen zu gewährleisten, wurde das `GenericCRUDProtocol` entwickelt. Dieses Protokoll abstrahiert die grundlegenden Datenbankoperationen wie das Erstellen (Create), Abrufen (Read), Aktualisieren (Update) und Löschen (Delete), auch bekannt als *CRUD*-Operationen. Das Protokoll ist generisch gestaltet, sodass es für jedes Datenmodell, das in der Datenbank gespeichert wird, wiederverwendet werden kann (vgl. Abb. 10a).

Das `GenericCRUDProtocol` sieht folgendermassen aus:

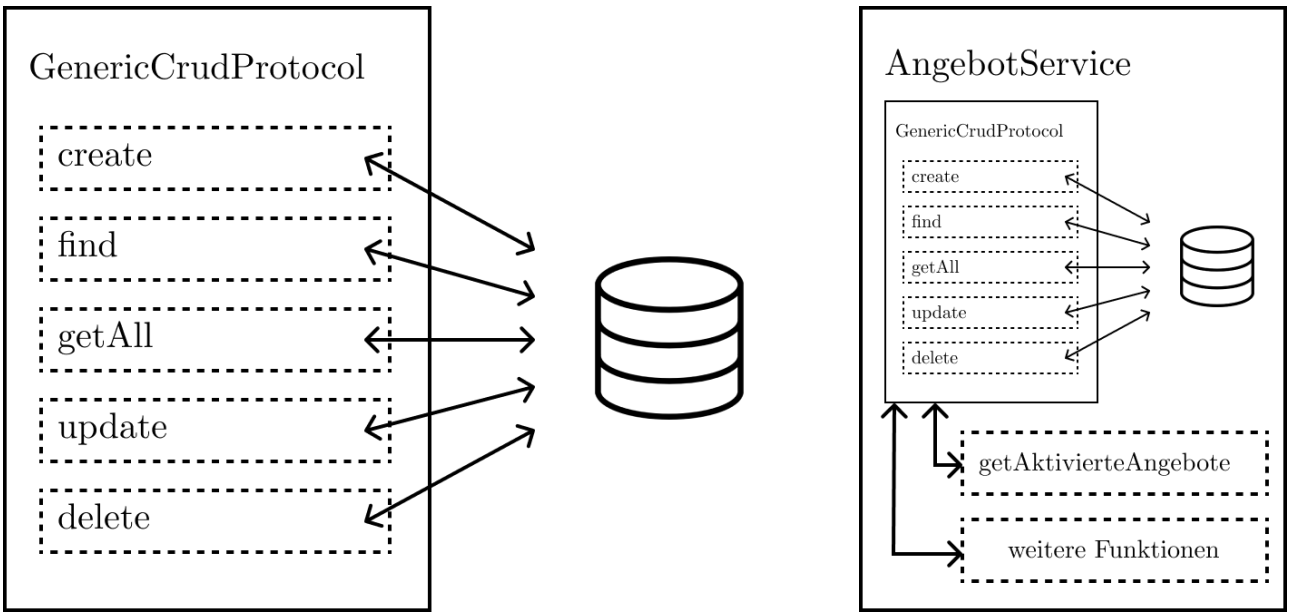
Code:

```
// Ein generisches Protokoll für CRUD-Operationen (Create, Read, Update, Delete) für
ein Model:
protocol GenericCRUDProtocol where ModelType: Model, ModelType.IDValue:
LosslessStringConvertible {
    // Der assoziierte Typ des Modells, das dieses Protokoll implementiert:
    associatedtype ModelType

    // Erstellt ein neues Modell in der Datenbank:
    static func create(_ req: Request, model: ModelType) async throws {
        try await model.create(on: req.db)
    }

    // Findet ein Modell anhand der ID in der Datenbank:
    static func find(_ req: Request, id: UUID) async throws -> ModelType? {
        guard let model = try await ModelType.find(id, on: req.db) else {
            throw Abort(.notFound, reason: "Modell nicht gefunden.")
        }
        return model
    }

    // Gibt alle Modelle aus der Datenbank zurück:
    static func getAll(_ req: Request) async throws -> [ModelType] {
        return try await ModelType.query(on: req.db).all()
    }

    // Aktualisiert ein bestehendes Modell in der Datenbank:
    static func update(_ req: Request, id: UUID, updatedModel: ModelType) async throws {
        guard let existingModel = try await find(req, id: id) else {
            throw Abort(.notFound, reason: "Modell nicht gefunden.")
        }
        try await existingModel.update(on: req.db)
    }

    // Löscht ein Modell aus der Datenbank:
    static func delete(_ req: Request, id: UUID) async throws {
        guard let model = try await find(req, id: id) else {
            throw Abort(.notFound, reason: "Modell nicht gefunden.")
        }
        try await model.delete(on: req.db)
    }
}
```

Das vom Verfasser entwickelte Protokoll definiert Methoden zur Erstellung, zum Abrufen eines einzelnen Eintrags oder aller Einträge, zur Aktualisierung und zum Löschen eines Eintrags. Dadurch muss nicht für jedes neue Modell erneut derselbe Code geschrieben werden, sondern es kann das generische Protokoll verwendet werden. Dafür wird für jede Datenstruktur eine Service-Klasse erstellt, basierend auf dem `GenericCrudProtocol` (vgl. Abb. 10b). Beispielsweise ein `AngebotService`:

Code:

```
// Service für CRUD-Operationen und spezifische Logik für das AngebotModel:
struct AngebotService: GenericCRUDProtocol {
    // Legt fest, dass dieses Service-Angebot für das Modell "AngebotModel" gilt:
    typealias ModelType = AngebotModel

    // Methode, um nur aktivierte Angebote aus der Datenbank zu holen:
    static func getAktivierteAngebote(req: Request) async throws -> [AngebotModel] {
        return try await getAll(req).filter { $0.aktiviert }
    }
}
```



(a) Aufbau des Protokolls

(b) Anwendung des Protokolls im AngebotService

Abb. 10: Aufbau und Anwendung des GenericCRUDProtocol

2.2.3 Geteilte Codebasis

Die Implementierung von Server und App in Swift bietet den Vorteil, dass Teile des Codes effizient geteilt werden können. In diesem Projekt umfassen die geteilten Komponenten hauptsächlich die Datenmodelle (z.B. `Angebot`), HTTP-Schnittstellen und Hilfs-Funktionen. Dank einer geteilten Codebasis muss der Code für die Kommunikation zwischen Server und Gerät nur einmal geschrieben werden (vgl. Abb. 11). Dieses gemeinsame Swift-Paket wurde auf Grundlage einer Präsentation von Martin Lasek implementiert [38]. Das Paket wird über GitHub [39] verwaltet und wurde mit dem Swift Package Manager [40] in die Anwendungen integriert. Der App-Code, der Server-Code und die geteilte Codebasis sind einheitlich nach dem *Documentation Compiler (DocC)*-Standard dokumentiert [41].

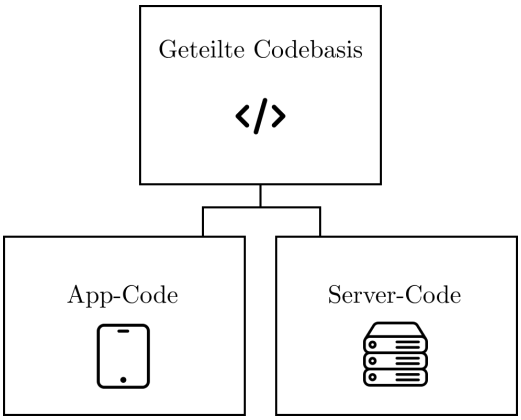


Abb. 11: Aufbau der geteilten Codebasis

2.2.3.1 Models

Die Datenmodelle, wie `Angebot`, müssen nur einmal definiert werden und können sowohl im Server- als auch im App-Kontext verwendet werden. Um beispielsweise einem Modell ein Argument hinzuzufügen, ist es ausreichend, dies einmal anzupassen; sowohl der Server als auch die App übernehmen die Änderung automatisch.

Um die Daten der Models auch in der Datenbank speichern zu können und mit Fluent zu arbeiten, wurden separate Model-Objekte (z.B. das `AngebotModel` für `Angebot`) für die Datenbank erstellt und eine Funktion, um zwischen den normalen und den Datenbank-Modellen hin und her zu konvertieren, implementiert:

Code:

```
extension Angebot: Content {
    // Erstellt ein Angebot-Objekt aus einem Fluent AngebotModel-Objekt:
    static func create(from angebotModel: AngebotModel) -> Angebot {
        return Angebot(
            id: angebotModel.id,
            titel: angebotModel.titel,
            beschreibung: angebotModel.beschreibung,
            aktiviert: angebotModel.aktiviert
        )
    }
}

// Nutzung:
let angebotModel = AngebotModel(...) // Ein bestehendes AngebotModel-Objekt.
let angebot = Angebot.create(from: angebotModel) // Wandelt AngebotModel in ein Angebot um.
```

2.2.3.2 HTTP-Schnittstellen

Die HTTP-Schnittstellen sind auf Basis eines Artikels von Antoine van der Lee [42] aufgebaut. Hierfür wurde ein Protokoll namens `RequestProtocol` erstellt, welches die Struktur der HTTP-Anfragen festlegt.
Zusätzlich wurde das `RouteProtocol` eingeführt, um die Routenstruktur klar zu definieren. Dieser Code befindet sich in der geteilten Codebasis:

Code:

```
public protocol RequestProtocol {
    // Der Typ des Anfragekörpers (z.B. ein Angebot, dass zu erstellen ist):
    associatedtype RequestBody: Codable

    // Der Typ des Antwortkörpers (standardmässig HTTPStatus):
    associatedtype ResponseBody: Codable = HTTPStatus

    // Die HTTP-Methode für die Anfrage (GET oder POST):
    var method: HTTPMethod { get }

    // Der Pfad (URL-Endpoint), auf den die Anfrage abzielt, z.B. "create":
    var path: String { get }
}

public protocol RouteProtocol {
    // Der Pfad der Route, z.B. "angebote":
    static var path: String { get }
}
```

Anhand dieser Protokolle wurden verschiedene Routen erstellt, die unterschiedliche Anfragen enthalten, was eine klare und strukturierte Organisation der Codebasis ermöglicht. Der folgende Code-Ausschnitt zeigt die `AngebotRoute` und deren HTTP-Anfragen:

Code:

```
// Definiert die Route und zugehörige Anfragen für Angebote mithilfe des RouteProtocol:
public class AngebotRoute: RouteProtocol {
    public static var path = "angebote"
    public init() {}

    // Route für das Erstellen eines neuen Angebots mithilfe des RequestProtocol:
    public struct CreateRequest: RequestProtocol {
        public let method = HTTPMethod.post; public let path = "create"

        // Anfragekörper für das Erstellen eines Angebots:
        public struct CreateRequestBody: Codable {
            public let titel: String
            public let beschreibung: String

            public init(titel: String, beschreibung: String) {
                self.titel = titel; self.beschreibung = beschreibung
            }
        }
        public init() {}
    }

    // Route für das Löschen eines Angebots:
    public struct DeleteRequest: RequestProtocol {
        public let method = HTTPMethod.delete; public let path = "delete"

        // Anfragekörper für das Löschen eines Angebots:
        public struct DeleteRequestBody: Codable {
            public let id: UUID // Die ID des zu löschenden Angebots.

            public init(id: UUID) { self.id = id }
        }
        public init() {}
    }

// Hier folgen weitere Anfragen, z.B. "AngebotÜbernehmen" oder "AngebotBeenden"
}
```

Auf dem Server kann damit spezifiziert werden, wie jede Anfrage verarbeitet werden sollte. Dazu wurde ein `ReceivableRequest`-Protokoll entwickelt, welches in der Codebasis des Servers befindet. Das Protokoll ermöglicht es, für jede in der gemeinsamen Codebasis registrierte Anfrage eine `handle`-Methode zu definieren, die festlegt, wie die Anfrage auf dem Server verarbeitet werden soll. Ausserdem besitzt das Protokoll eine Methode zur Registrierung der Anfrage bei Vapor:

Code:

```
protocol ReceivableRequest: RequestProtocol {
    // Die Hauptlogik zur Verarbeitung der Anfrage, für jede Anfrage individuell:
    func handle(req: Request, body: RequestBody) async throws -> ResponseBody

    // Registriert die Anfrage bei Vapor, sodass sie z.B. unter "angebote/create"
aufrufbar ist:
    func registerRequest(_ routes: RoutesBuilder) {
        let pathComponent = PathComponent(stringLiteral: self.path)

        switch method {
        case .get:
            routes.get(pathComponent) { req in
                let body = try req.content.decode(RequestBody.self)
                return try await handle(req: req, body: body)
            }
        case .post:
            routes.post(pathComponent) { req in
                let body = try req.content.decode(RequestBody.self)
                return try await handle(req: req, body: body)
            }
        }
    }
}
```

Das `ReceivableRequest` kann nun jeder bereits definierten Anfrage hinzugefügt werden. Dabei lässt sich in der `handle`-Methode spezifizieren, wie die Anfrage verarbeitet werden soll, beispielsweise für die Angebot-Erstellen-Anfrage:

Code:

```
// Erweiterung, um die CreateRequest-Route zu einer empfangsbereiten Anfrage zu machen
(ReceivableRequest):
extension AngebotRoute.CreateRequest: ReceivableRequest {
    // Implementiert die handle-Methode, die beschreibt, wie die Anfrage verarbeitet wird:
    func handle(req: Vapor.Request, body: CreateRequestBody) async throws ->
Shared.HTTPStatus {
        // Nutzt den AngebotService, um ein Angebot zu erstellen.
        try await AngebotService.create(req, from: body)
        return .ok
    }
}
```

In der App konnten darauf im ViewModel die verschiedenen Anfragen implementiert werden. Dazu wurde direkt auf die Anfragen aus der geteilten Codebasis zugegriffen:

Code:

```
// Funktion im ViewModel der App zum Erstellen eines Angebots:
func createAngebot(titel: String, beschreibung: String) {
    Task {
        do {
            // Erstellt eine Instanz der CreateRequest-Route aus der geteilten Codebasis.
            let createRequest = AngebotRoute.CreateRequest()

            // Baut den Anfragekörper mit den übergebenen Parametern:
            let body = AngebotRoute.CreateRequest.CreateRequestBody(titel: titel,
beschreibung: beschreibung)

            // Sendet die Anfrage über den NetworkClient und wartet auf die Antwort:
            let _: HTTPStatus = try await NetworkClient.shared.send(
                createRequest,
                routePath: AngebotRoute.path,
                body: body
            )
        }
    }
}
```

2.2.3.3 Hilfs-Funktionen

Zuletzt wurden, zusätzlich zu den Modellen und Schnittstellen, gemeinsame Hilfs-Funktionen, wie etwa einen `JSONDecoder` implementiert, die sicherstellen, dass die Modelle sowohl in der App als auch auf dem Server korrekt kodiert und dekodiert werden:

Code:

```
// Erstellung eines einheitlichen JSONDecoder:
public extension JSONDecoder {
    static var shared: JSONDecoder {
        let decoder = JSONDecoder()

        // Legt fest, dass Datumswerte im ISO 8601-Format dekodiert werden.
        decoder.dateDecodingStrategy = .iso8601
        return decoder // Gibt den konfigurierten Decoder zurück.
    }
}

// Nutzung, in der App oder auf dem Server:
let jsonDecoder = JSONDecoder.shared
```

2.3 App-Design

Die App, die auf den iPads an den verschiedenen Angeboten laufen wird, soll es den Kindern ermöglichen, sich durch das Scannen ihrer Ausweise ein- und auszuchecken. Gleichzeitig muss die App den Betreuungspersonen die Möglichkeit bieten, durch das Scannen ihrer Ausweise auf zusätzliche Funktionen zuzugreifen, wie das Übernehmen oder Beenden von Angeboten oder das Suchen von Kindern.
Die Gestaltung der App orientierte sich sowohl an den Apple Human Interface Guidelines [43] als auch an den Google Building for Kids Guidelines [44].
Ein zentrales Prinzip bei der Gestaltung des *User Interface (UI)* und der *User Experience (UX)* ist die Unterstützung der Betreuungspersonen. Die App soll ihre Arbeit erleichtern und keine zusätzliche Hürde darstellen. Die Benutzeroberfläche muss intuitiv und einfach sein, mit möglichst wenig Interaktion. Laut den Designprinzipien von Google sollte unnötige Komplexität vermieden werden. Je weniger eine Betreuungsperson klicken oder navigieren muss, desto mehr Arbeit wird ihr abgenommen. Eine durchdachte Gestaltung sorgt dafür, dass sich die Betreuungspersonen schnell zurechtfinden.

Die Kinder müssen die App nicht berühren; sie scannen nur ihre Ausweise. Dies spart Zeit und verhindert verschmutzte iPads. Sobald sich ein Kind ein- oder auscheckt, gibt das System sofort klare visuelle und auditive Rückmeldungen. Dies ermöglicht es, jeder Klasse nur eine kurze Einführung zu geben, ohne dass jedes Kind individuelle Anweisungen benötigt.
Wichtig ist ausserdem, dass die Kinder das System freiwillig und aktiv anwenden. Dies kann laut Google-Designprinzipien durch Elemente wie Konfetti, Geräusche oder Animationen geschehen. Die Nutzung des Systems ist für die Kinder dadurch positiv assoziiert und hält sie motiviert.

2.4 Iterative Entwicklung

Bei der Entwicklung und dem Design der App wurde der Prozess der iterativen Entwicklung genutzt. Unter den englischen Synonymen «Iterative Design» oder «Rapid Prototyping» ist ein Entwicklungsprozess zu verstehen, bei dem ein Prototyp erstellt, getestet und evaluiert wird, um daraufhin sofort weiterentwickelt zu werden [45]. Das Ziel ist, das Produkt so früh wie möglich an die Nutzer zu bringen, um schnell Feedback zu erhalten. Dieses Konzept wurde in diesem Projekt eingesetzt. Durch das Beobachten und das Einholen von Feedback von Kindern und Betreuungspersonen konnten Unklarheiten schnell identifiziert und behoben werden.
Aus diesem Grund wurde das System zuerst in Pilotphasen getestet, um die App zügig zu verbessern und anzupassen. Dies betraf sowohl das App-Design, bei dem beispielsweise die Platzierungen und Schriftgrössen angepasst wurden, um die Klarheit zu erhöhen, als auch Anpassungen im Backend, wie die Umstellung von HTTP-Anfragen auf WebSockets für eine effizientere Datenübertragung (vgl. Kap. 3.4).

2.5 Datenschutz

Bei der Einführung eines digitalen Systems stellt sich die Frage nach dem Datenschutz: Werden die Kinder getrackt? Was passiert mit ihren Daten? Diese berechtigten Bedenken könnten besonders bei Eltern im Zuge der fortschreitenden Digitalisierung laut werden. Ein Beispiel dafür ist eine kürzlich erschienene Meldung in den Medien: «Eine Zürcher Primarschule will Hortkinder mit Trackern ausstatten, um sie zu überwachen», berichtet die NZZ [46]. Laut dieser Meldung sollten Kinder im Hort Letten magnetische Armbänder mit einem Bluetooth-System tragen, die es den Betreuungspersonen ermöglichen, ihren Standort auf dem Schulgelände zu verfolgen. Das Projekt löste heftige Kritik aus und wurde nach Medienberichten und einem Elternabend in Birmensdorf vorerst auf Eis gelegt. Im Tages-Anzeiger wurde die kantonale Datenschutzbeauftragte, Dominika Blonski, zitiert: «Eine dauerhafte Lokalisierung von Schülerinnen und Schülern scheine für die Betreuung von Kindern nicht erforderlich zu sein.» [47]

Um derartige Situationen zu vermeiden, wurde das System der Schule Freilager von Anfang an mit einem klaren Fokus auf den Schutz der Privatsphäre entwickelt. Statt auf Trackingsysteme wie Bluetooth-Armbänder zu setzen, wurden einfache, batterielose Hausausweise verwendet. Diese müssen von den Kindern aktiv gescannt werden, sodass sie nicht permanent überwacht werden. Die einzige Information, die die Betreuungspersonen erhalten, ist das letzte Angebot, das ein Kind besucht hat, jedoch ohne genaue Ortung des Kindes selbst. Das System weiss nicht, wo sich ein Angebot wie die «Turnhalle» befindet, sondern nur, dass dieses existiert.
Trotzdem werden notwendige Daten erfasst, nämlich wer sich wann in ein Angebot ein- und ausgecheckt hat. Um den Datenschutz zu gewährleisten, wurde eine On-Premises-Lösung entwickelt, bei der alle Daten lokal auf dem Server innerhalb des Schulnetzwerks bleiben. Keine Daten verlassen das Schulgebäude. Darüber hinaus werden die Check-in-Daten täglich um 1 Uhr nachts gelöscht, da es unverhältnismässig wäre, Check-in-Daten über längere Zeit zu speichern.

Diese Vorgehensweise entspricht den Vorgaben von § 11 IDG, nach welchem Datenbearbeitungssysteme so gestaltet werden müssen, dass nur notwendige Personendaten anfallen, und diese, sobald nicht mehr benötigt, gelöscht, anonymisiert oder pseudonymisiert werden.
Das neue System ersetzt das bisherige Magnettafelsystem und das manuelle Abhaken der Anwesenheitsliste lediglich digital. Es werden nur die bereits existierenden und notwendigen Daten verarbeitet und nur so lange gespeichert, wie es wirklich erforderlich ist. Dies steht mit § 7 IDG im Einklang, wonach Informationen durch angemessene organisatorische und technische Massnahmen geschützt werden müssen, um sicherzustellen, dass sie richtig, vollständig und nur bei Bedarf verfügbar sind [2].

2.6 Mögliche Kritikpunkte

Um potenzielle Unklarheiten hinsichtlich der gewählten Methodik und deren Umsetzung auszuräumen, werden im Folgenden einige denkbare Kritikpunkte behandelt.

2.6.1 Technologieauswahl: Optische Codes vs. RFID/NFC

Ein möglicher Kritikpunkt an der gewählten Technologie ist die Entscheidung, optische Codes (Barcodes/QR-Codes) anstelle von RFID- oder NFC-Tags zu verwenden. RFID- und NFC-Tags bieten langfristig eine höhere Flexibilität und könnten in Situationen mit mehr Nutzern effizienter sein, da sie keine aktive Handlung des Benutzers erfordern und automatisch gescannt werden können. Insbesondere RFID-Tags ermöglichen ein passives Tracking über grössere Entfernungen, was das Ein- und Auschecken der Kinder noch einfacher gestalten könnte.

Die Entscheidung für optische Codes lässt sich jedoch durch mehrere Faktoren rechtfertigen: Sie sind kostengünstiger, da keine teuren RFID-Scanner benötigt werden. Beschädigte oder verlorene Karten können schnell und praktisch kostenlos mit einem handelsüblichen Kartendrucker ersetzt werden. Zudem sind optische Codes leicht zu generieren und können in der Regel mit bereits vorhandener Hardware (wie z.B. iPads mit Kamera) genutzt werden, was die Gesamtkosten des Systems erheblich senkt. Auch aus Datenschutzgründen sind QR-Codes vorteilhaft, da kein automatisiertes Tracking stattfindet. Die Kinder behalten die Kontrolle über das Ein- und Auschecken.

2.6.2 Kompatibilität mit Non-Apple-Geräten

Zu monieren wäre weiter die fehlende Kompatibilität mit Non-Apple-Geräten. Da das System in SwiftUI programmiert wurde, ist die Nutzung auf anderen Geräten wie Android-Tablets unmöglich. Dies könnte zukünftige Erweiterungen oder die Einbindung anderer Schulen, die möglicherweise andere Geräte nutzen, erschweren. Es fragt sich daher, ob eine plattformunabhängige Weblösung vorteilhafter wäre. Eine Webanwendung hätte den Vorteil, dass sie auf jedem Gerät mit einem Browser läuft und nicht von bestimmten Betriebssystemen abhängig ist.

Für eine native App sprechen mehrere Vorteile. Zunächst bietet eine native App eine deutlich höhere Leistung und Zuverlässigkeit im Vergleich zu einer Webanwendung. Da sie speziell für die Hardware entwickelt wurde, kann sie auf die gerätespezifischen Funktionen wie Kamera und Speicher direkt zugreifen, was eine schnellere und flüssigere Nutzung ermöglicht. Webanwendungen erfordern oft zusätzliche Anpassungen, um auf verschiedenen Geräten und Browsern reibungslos zu funktionieren, was den Entwicklungsaufwand massiv erhöht.
Sodann spricht die Benutzerfreundlichkeit für die native App. Die App kann direkt gestartet werden, ohne dass ein Browser geöffnet und durch eine Web-Oberfläche navigiert werden

muss. Dies spart Zeit und macht die Anwendung für die Betreuungspersonen einfacher und intuitiver. Webanwendungen können komplizierter wirken, insbesondere wenn auf verschiedenen Geräten und Browsern eine konsistente Erfahrung gewährleistet werden soll. Schliesslich muss auch abgewogen werden, ob eine stabile und optimierte Lösung für Apple-Geräte nicht besser ist als eine halb-stabile Lösung, die zwar auf allen Geräten funktioniert, aber in puncto Zuverlässigkeit und Benutzerfreundlichkeit Kompromisse eingeht. Im Hinblick auf eine mögliche Ausweitung des Systems sind Apple-Geräte in vielen Schulen bereits weit verbreitet. Eine native App bietet hier die Möglichkeit, das volle Potenzial dieser Geräte auszuschöpfen. Eine plattformübergreifende Weblösung würde hingegen wahrscheinlich in ihrer Leistungsfähigkeit und Stabilität hinter einer nativen App zurückbleiben, was im täglichen Einsatz zu Schwierigkeiten führen könnte.

2.6.3 Lokales Netzwerk und On-Premises-Lösung vs. Schulnetzwerk und Cloud-Lösung

Auch die Entscheidung, ein separates lokales Netzwerk und eine On-Premises-Lösung zu verwenden, kann hinterfragt werden. Die Einrichtung eines eigenen Netzwerks erfordert regelmässige Wartung und birgt das Risiko von Ausfällen. Eine Integration in das bestehende Schulnetzwerk wäre stabiler und einfacher zu verwalten, da dieses bereits existiert, genutzt und verwaltet wird. Durch die Anbindung an das zentrale Netzwerk bestünde zudem die Möglichkeit, statt einer lokalen On-Premises-Lösung eine Cloud-Lösung zu implementieren. Das hiesse, einen Cloud-Server statt den Mac Mini zu nutzen, was die Wartung und Verwaltung der Hardware vor Ort minimiert und eine bessere Skalierbarkeit für eventuell weitere Schulen geboten hätte.

Allerdings wurde in diesem Projekt die Entscheidung für ein separates lokales Netzwerk und eine On-Premises-Lösung bewusst getroffen. Ein eigenes Netzwerk verringert den bürokratischen Aufwand, da es keine Änderungen am zentralen Schulnetzwerk erfordert, das in der Regel durch Firewalls geschützt ist und für jede Modifikation Genehmigungen benötigt. Dies hätte die Einführung des Systems massiv verzögert. Ein lokales Netzwerk ermöglicht eine schnelle und unkomplizierte Inbetriebnahme, ohne den laufenden Schulbetrieb zu beeinträchtigen. Zudem bietet die On-Premises-Lösung volle Kontrolle über die Daten, was Datenschutzrisiken vermeidet, die mit der externen Speicherung in der Cloud verbunden wären. Es bleibt sichergestellt, dass sensible Informationen der Kinder nur innerhalb der Schule verwaltet werden.

2.6.4 Swift als Server-Programmiersprache

Ein weiterer Kritikpunkt könnte die Wahl von Swift als Programmiersprache für den Server sein, da Swift vor allem für die Entwicklung von iOS- und macOS-Anwendungen bekannt ist und weniger für Serveranwendungen. Typische Server-Programmiersprachen wie Python, Java oder Node.js haben eine grössere Community und bieten umfangreichere Libraries sowie eine bessere Dokumentation für Serverentwicklungen.

Die Wahl von Swift bringt jedoch einen entscheidenden Vorteil mit sich: Es besteht die Möglichkeit, Code zwischen der App und dem Server zu teilen. Dadurch müssen Datenstrukturen, Modelle und Logiken nur einmal definiert werden, was das Risiko von Fehlern verringert, die durch doppelte Implementierungen in unterschiedlichen Sprachen entstehen könnten. Diese gemeinsame Codebasis macht die Entwicklung effizienter und konsistenter, da Änderungen nur an einer Stelle vorgenommen werden müssen und direkt in beiden Bereichen wirksam werden. Dazu bietet Swift selbst eine hervorragende Leistung und Sicherheit. Dank der statischen Typisierung werden viele Fehler bereits zur Compile-Zeit erkannt, was die Robustheit des Systems erhöht.

2.6.5 Datenschutz und Effizienz des Systems

Schliesslich kann die Einführung eines digitalen Systems zur Erfassung der Aufenthaltsorte der Kinder als solches hinterfragt werden. Es kann argumentiert werden, dass ein solches System unverhältnismässig sei und dass die bisherige Lösung, bestehend aus einer Magnettafel und dem Abhaken einer Anwesenheitsliste, ausreichend wäre. Insbesondere könnte der Vorwurf des «Trackings» aufkommen, da die Check-in-Daten der Kinder nun nicht mehr manuell erfasst werden.

Nochmals zu betonen ist, dass das System kein «Tracking» im Sinne einer aktiven, umfassenden Überwachung betreibt. Das System ersetzt lediglich die bestehende Magnettafel und das Abhaken der Anwesenheitsliste in einer modernen, digitalen Form. Die Funktionalität bleibt dieselbe: Es wird nur gespeichert, welches Kind sich wann bei einem Angebot ein- oder ausgecheckt hat. Diese Information wurde bereits vorher erfasst – nur eben manuell. Daher ist das System nicht unverhältnismässig, sondern eine notwendige und zeitgemässe Weiterentwicklung. Zudem reicht das analoge System in grösseren Tagesschulen nicht mehr aus, da es Betreuungspersonen stark mit administrativen Aufgaben belastet. Der Bedarf nach einer effizienteren Lösung bestand in der Schule Freilager eindeutig. Das digitale System automatisiert diese Prozesse und entlastet somit das Personal, sodass mehr Zeit für die eigentliche Betreuung bleibt.
Ein weiterer entscheidender Faktor ist, dass die Schülerzahlen in vielen Schulen stetig steigen. Die Verwaltung von immer mehr Kindern und deren Bewegungen innerhalb der Schule wird mit einem analogen System zunehmend komplizierter und fehleranfälliger. Eine digitale Lösung wird langfristig unverzichtbar, um den wachsenden Anforderungen gerecht zu werden.
Das System entspricht den Vorgaben des IDG und der IDV. Es werden nur die notwendigsten Daten – die Ein- und Auscheckzeiten – erfasst, lokal gespeichert und regelmässig gelöscht. Keine sensiblen Informationen verlassen das Schulgebäude, und es findet kein Tracking statt.

3 Ergebnis

3.1 ORBITS

Das System erhielt den Namen ORBITS: **Or**ganisation der **B**etreuung **i**n **T**agess**s**chulen. Der Name spiegelt die Kernaufgabe des Systems wider: Die Sicherstellung einer strukturierten und zuverlässigen Organisation der Mittagsbetreuung an Tagesschulen. ORBITS soll den Überblick über die Verteilung der Kinder erleichtern, administrative Prozesse digitalisieren und damit die Effizienz sowie Sicherheit erhöhen.
Das Ziel von ORBITS war es nicht nur, die administrativen Aufgaben für Betreuungspersonen zu vereinfachen, sondern auch eine solide Grundlage zu schaffen, die flexibel an wachsende Anforderungen angepasst werden kann. Mit dieser Anwendung wurde versucht, einen Schritt in Richtung Digitalisierung und Zukunftsfähigkeit der Betreuung in Tagesschulen zu gehen.
Darüber hinaus bot der Name ORBITS eine symbolische Bedeutung: Ähnlich wie Planeten ihre Bahnen ziehen, sollen die Kinder in ihren Tagesabläufen «umherkreisen», wobei das System als die zentrale Achse agiert, die Überblick und Stabilität bietet.

3.2 Erster Einsatz in der Schule Freilager

Nach Abschluss der Entwicklungs- und Implementierungsarbeiten war die erste Version der App einsatzbereit. Der Server wurde eingerichtet, das lokale Netzwerk aufgebaut, und die App auf den iPads installiert. Die notwendigen Daten, wie die Kinderprofile und die Anwesenheitsliste, wurden in das System eingepflegt. Betreuungspersonen erhielten eine Einführung in das neue System und Eltern wurden vorab per E-Mail informiert. Zwei Pilotklassen mit insgesamt 45 Kindern wurden in einer kurzen Einführung mit der Funktionsweise der App vertraut gemacht. Die iPads wurden an der Mensa, der Bibliothek, dem Eingang zum Aussenbereich und der Turnhalle positioniert.
Beim ersten Einsatz verlief der Start planmässig. Die Kinder konnten ihre Hausausweise an den iPads scannen, um sich in Angebote ein- oder auszuchecken, und die Platzanzahlen wurden auf allen Geräten in Echtzeit synchronisiert. Nach etwa zwei Minuten kam es jedoch zu einem vollständigen Systemausfall. Die Anfragen an den Server wurden zunehmend langsamer verarbeitet, bis schliesslich keine Daten mehr verarbeitet wurden.
Zur Behebung des Problems wurde der Server neu gestartet, und die Anzahl der verfügbaren Angebote auf zwei – Mensa und Bibliothek – reduziert. Mit dieser eingeschränkten Konfiguration lief das System stabil und ohne Verzögerungen.
Der Testlauf mit den Pilotklassen lieferte wertvolle Erkenntnisse über die praktische Anwendung des Systems. Die genaue Ursache für den Serverausfall konnte jedoch zunächst nicht identifiziert werden.

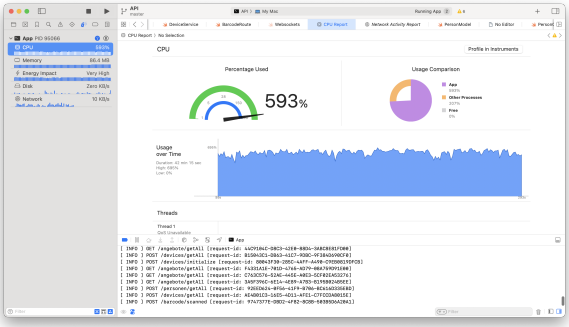
3.2.1 Mögliche Fehlerquellen

Bei der Überprüfung des Servers stellte sich heraus, dass kein Programmierfehler vorlag. Ein solcher Fehler hätte zudem zu einem sofortigen Ausfall geführt und nicht zu einem allmählichen Verlangsamen des Systems.
Möglich war eine Überlastung des Netzwerks. Da jedes Gerät jede Sekunde eine HTTP-Anfrage an den Server sendete, um alle aktuellen Daten abzurufen, könnte dies die Bandbreite erschöpft haben, sodass keine weiteren Daten mehr übertragen werden konnten.
Diese übermässige Anzahl an Anfragen kann noch weitere negative Auswirkungen hervorrufen. Zum einen wäre denkbar, dass sie zu einer Überlastung der Datenbank führen, da zu viele Abfragen gleichzeitig verarbeitet werden müssen, was wichtige Serverressourcen beansprucht.

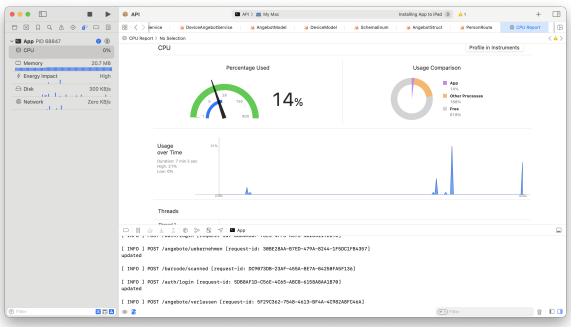
Zum anderen wäre es möglich, dass die *Central Processing Unit (CPU)* des Servers überlastet ist, da die Hardware gegebenenfalls nicht dafür ausgelegt ist, eine solch hohe Anzahl an Anfragen gleichzeitig zu verarbeiten. Jede eingehende HTTP-Anfrage erfordert, dass der Server die Anfrage analysiert, den entsprechenden Code ausführt und eine Antwort generiert. Wenn viele Anfragen in kurzer Zeit eintreffen, müssen diese Aufgaben parallel ausgeführt werden, was die CPU stark beansprucht.
Das Problem konnte reproduziert werden, indem mehrere iPad-Simulatoren über einen Zeitraum von mehreren Minuten im Sekundentakt Anfragen an den Server sendeten. Dabei zeigte sich erneut, dass die Bearbeitungszeit der Anfragen kontinuierlich zunahm, bis der Server schliesslich nicht mehr reagierte. Um festzustellen, ob die Ursache in der Datenbank oder der CPU-Auslastung liegt, wurde der Test mit Anfragen wiederholt, die keine Datenbankaktionen erforderten. Da das Problem weiterhin bestand, konnte die Datenbank als Fehlerquelle ausgeschlossen werden.
Durch die Nutzung der Debug-Tools von Xcode konnte sowohl die CPU- als auch die Netzwerkauslastung überwacht werden. Während das Netzwerk noch ausreichend Kapazitäten aufwies, war die CPU stark ausgelastet. Das Problem war somit identifiziert: Die vielen HTTP-Anfragen führten zu einer CPU-Überlastung des Servers (vgl. Abb. 12a).

3.2.2 Umstellung auf WebSockets

Zwei mögliche Lösungsansätze wurden in Betracht gezogen: Einerseits die Aufrüstung der Hardware durch den Einsatz eines leistungsfähigeren Servers mit höherer CPU-Kapazität, andererseits die Optimierung der ineffizienten Methode der HTTP-Requests. Die zweite Option erschien als naheliegender Ansatz. Wie bereits in der Methodik beschrieben, eignen sich WebSockets besonders gut für Live-Datenübertragungen.
Daher wurde die Implementierung von WebSockets vorgenommen, sodass jedes iPad beim Start der App eine dauerhafte Verbindung zum Server herstellt. Statt im Sekundentakt HTTP-Requests zu senden, werden aktualisierte Daten nun nur dann an die iPads gesendet, wenn Änderungen vorliegen. Diese Anpassung reduzierte die CPU-Auslastung des Servers drastisch – von etwa 500-800% auf 0-30% (vgl. Abb. 12b).



(a) CPU-Auslastung des Servers vor der Umstellung auf WebSockets



(b) CPU-Auslastung des Servers nach der Umstellung auf WebSockets

Abb. 12: Screenshots der CPU-Leistung des Servers vor und nach der Umstellung auf WebSockets

3.3 Erweiterung auf sechs Klassen

Nachdem die Pilotphase mit zwei Klassen nach der Umstellung über mehrere Wochen stabil und erfolgreich verlaufen war, wurden dem System vier weitere Klassen hinzugefügt. Damit erhöhte sich die Anzahl Kinder im System auf rund 140. Sie nahmen das System ebenfalls mühelos auf, wohl auch, weil sie es bereits bei den ersten Pilotklassen beobachten konnten. Das System konnte die Steigerung der Nutzer ohne Einbussen in der CPU-Leistung verwalten. Folglich müssen die Betreuungspersonen bereits jetzt deutlich weniger Magnete bereitstellen. Das System wurde im Sinne der iterativen Entwicklung (vgl. Kap. 2.4) laufend angepasst, um das mündliche Feedback von Kindern, Betreuungspersonen und Administratoren umzusetzen. So wurde beispielsweise die Möglichkeit hinzugefügt, eine Liste von Kindern hochzuladen, damit diese nicht einzeln hinzugefügt werden müssen. Gleichzeitig gibt es noch immer Herausforderungen, die es zu lösen gilt, z.B. dass das Kamerafenster beim Starten von Angeboten gelegentlich schwarz bleibt und die App neu geöffnet werden muss.

3.4 Umstellung auf QR-Codes

Die Nutzbarkeit des Systems wurde zusätzlich durch die Umstellung auf QR-Codes verbessert. Zu Beginn war davon ausgegangen worden, dass der Scanner in der Bibliothek nur Barcodes scannen kann. Als später herausgefunden wurde, dass dieser auch QR-Codes unterstützt, wurden alle neuen Karten mit QR-Codes statt Barcodes gedruckt. Das Hinzufügen der QR-Code-Funktionalität in der App erforderte lediglich eine Änderung von einer Zeile. Die Betreuungspersonen berichteten, dass die QR-Codes sich als wesentlich benutzerfreundlicher erwiesen, da sie unabhängig vom genauen Abstand oder dem Winkel zur Kamera oft beim ersten Versuch korrekt erfasst werden.

3.5 Die finale App

3.5.1 Aufsetzen

Beim erstmaligen Aufstarten der App wird ausgewählt, in welchem Modus das Gerät funktionieren soll. Der erste und meistbenutzte Modus nennt sich «Angebot-Modus». Dieser erlaubt es Betreuungspersonen, Angebote zu starten, und Kindern, sich an diesen ein- und auszuchecken. Alle iPads, die an den Angeboten aufgestellt werden, nutzen diesen Modus. Der zweite Modus ist der «Info-Modus», der lediglich die aktiven Angebote mit den freien Plätzen anzeigt, beispielsweise für einen Apple-TV oder ein nicht verwendetes iPad. Der dritte Modus namens «Persönliches Gerät» erlaubt einer Betreuungsperson oder einem Administrator, sich zu Beginn einzuloggen und auf diesem Gerät immer eingeloggt zu bleiben. Es wird immer das persönliche Dashboard angezeigt. Dieser Modus kann auf dem persönlichen Apple-Gerät eines Administrators oder einer Betreuungsperson genutzt werden. Alle Geräte müssen dabei mit dem lokalen Netzwerk verbunden sein, um zu funktionieren. Beim erstmaligen Aufstarten der App wird ausgewählt, in welchem Modus das Gerät funktionieren soll (vgl. Abb. 13).

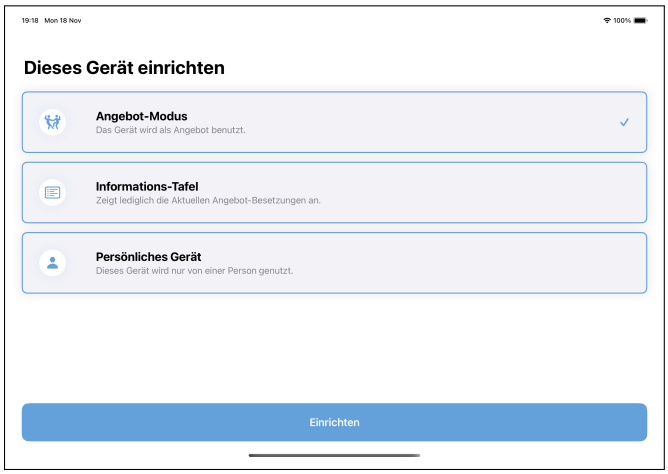


Abb. 13: App-Setup: Auswahl zwischen Angebot-Modus, Info-Modus und Persönliches Gerät

3.5.2 Täglicher Einsatz

Vor Beginn der Mittagspause holen die Betreuungspersonen pro Angebot ein eingerichtetes iPad aus dem Teamzimmer. Am jeweiligen Angebot öffnen sie die App, die zunächst nur einen Kamerastream und ein Symbol anzeigt, das dazu auffordert, den Hausausweis zu scannen. Sobald der QR-Code auf dem Hausausweis erkannt wurde, öffnet sich das Dashboard. Hier können Betreuungspersonen, neben anderen Funktionen, ein Angebot übernehmen. Dafür klicken sie auf das gewünschte Angebot (z.B. «Spaziergang»), das dadurch aktiviert wird (vgl. Abb. 15). Das Dashboard schliesst sich automatisch, und die Betreuungsperson befestigt das iPad an der vorgesehenen Wandhalterung. Weitere Betreuungspersonen können im Dashboard dem Angebot optional beitreten.

Abb. 14: Hausausweis mit QR-Code der Schule Freilager

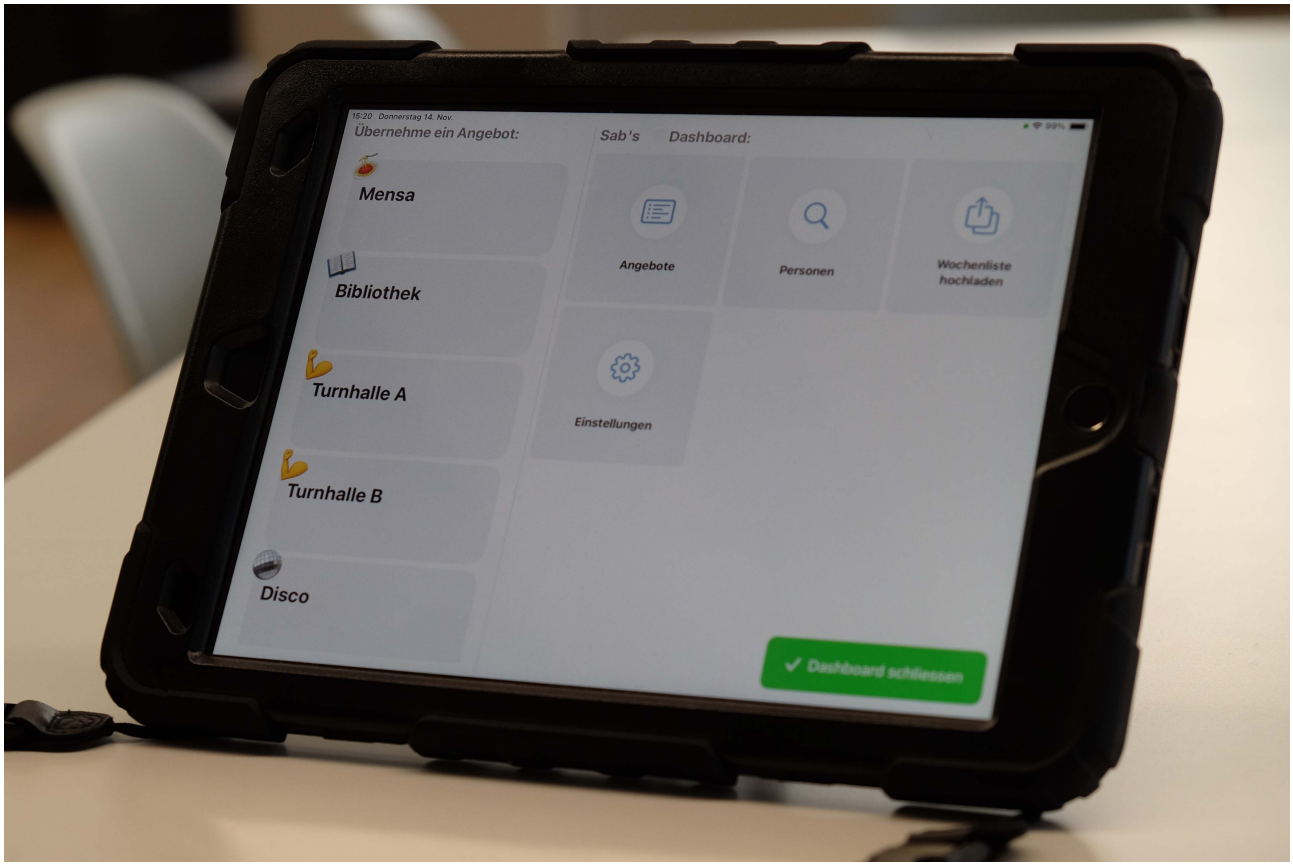
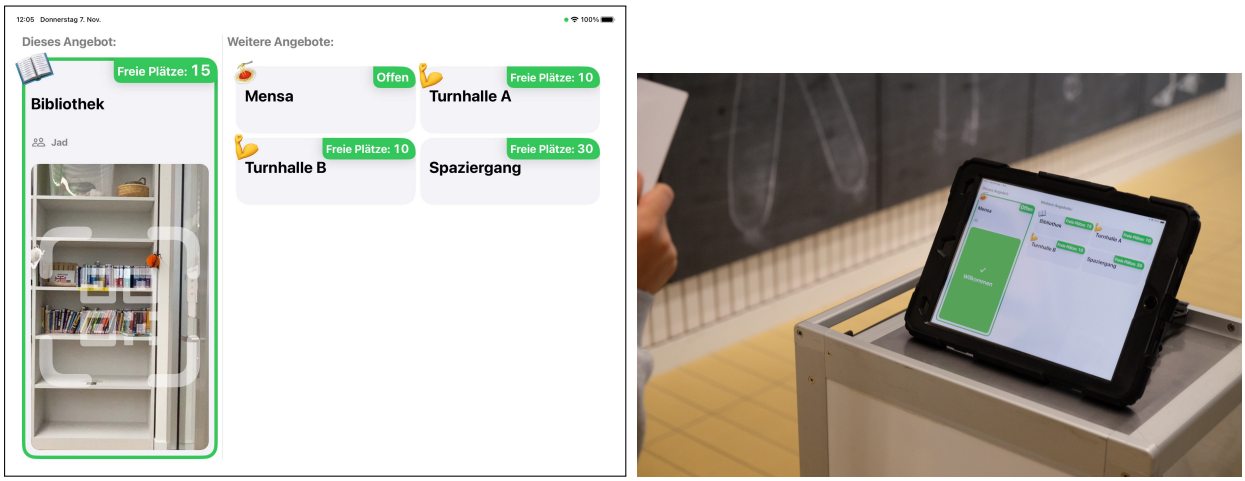


Abb. 15: Auswahl zur Aktivierung eines Angebots *(Namen zensiert)*

Sobald ein Angebot aktiviert wurde, ist das iPad bereit, die Kinder ein- und auszuchecken. Dafür wird ein kleines Kamerafenster angezeigt (vgl. Abb. 16a). Scannen die Kinder ihren Ausweis (vgl. Abb. 14), erscheint für kurze Zeit eine entsprechende Nachricht: «Willkommen, [Name]!» (vgl. Abb. 16b), wenn das Kind sich zum ersten Mal eincheckt; «Tschüss, [Name]!», wenn es sich mit dem zweiten Scannen beim selben Angebot auscheckt; oder «Checke dich zuerst beim [Angebot] aus», wenn das Kind vergessen hat, sich im vorherigen Angebot abzumelden. In diesem Fall muss das Kind zum letzten Angebot zurückkehren, um sich dort zunächst auszuchecken. Die App reagiert nicht auf Touch-Eingaben; Aktionen können nur mit den Ausweisen ausgeführt werden.



(a) Screenshot der Kinder-Ebene: Das «Angebot Bibliothek» ist aktiviert, Kinder können sich einloggen *(Namen zensiert)*

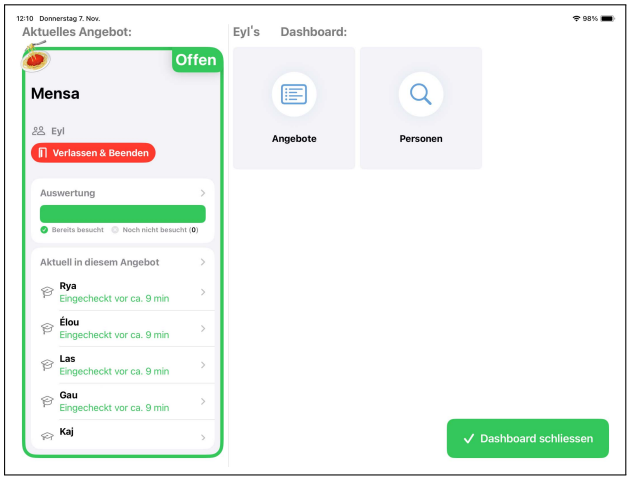
(b) Nach dem Scannen des Hausausweises wird eine persönliche Willkommensnachricht angezeigt *(Namen zensiert)*

Abb. 16: Die Kinder-Ebene im Einsatz: Angebote und Interaktion im Alltag

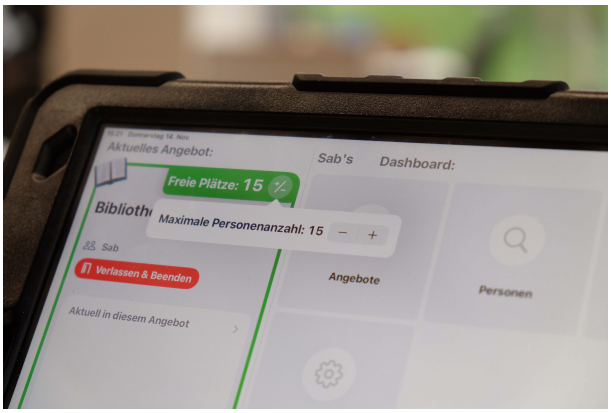
Neben dem Kamerafenster wird eine Übersicht zum aktuellen Angebot angezeigt: Titel, Beschreibung, Anzahl freier Plätze und die zuständige(n) Betreuungsperson(en). Ausserdem gibt es eine Liste mit anderen aktiven Angeboten und deren freien Plätzen, sodass die Kinder beim Auschecken sehen können, welche weiteren Angebote freie Plätze haben.
Sobald sich die Kinder auf die verschiedenen Angebote verteilt haben, können die Betreuungspersonen, die für die Mensa zuständig sind, durch erneutes Scannen ihres Ausweises im Dashboard die «Auswertung» aufrufen (vgl. Abb. 17a). Hier wird angezeigt, welche angemeldeten Kinder noch nicht zum Essen gekommen sind, einschliesslich ihres letzten oder aktuellen Aufenthaltsortes. Diese Funktion ersetzt das frühere manuelle Durchsehen der Mittagsliste und das anschliessende Suchen des Aufenthaltsortes auf der Magnettafel. Falls ein Kind am Ende der Pause noch nicht in der Mensa war, kann die Betreuungsperson es im entsprechenden Angebot daran erinnern, noch etwas zu essen. Sollte kein letztes Angebot angezeigt werden, bedeutet dies, dass das Kind sich noch nirgends eingeloggt hat. In solchen Fällen müssen die Betreuungspersonen klären, ob das Kind spontan krankgemeldet wurde oder sich anderweitig aufhält.

Während die Angebote laufen, stehen den Betreuungspersonen im Dashboard weitere Funktionen zur Verfügung. So können sie nachsehen, wo sich ein Kind zuletzt aufgehalten hat, wenn die Eltern es frühzeitig abholen. Ausserdem können sie die maximale Platzanzahl eines Angebots anpassen, z.B. wenn die Bibliothek mehr Kinder aufnehmen kann oder die Turnhalle überfüllt ist (vgl. Abb. 17b).
Am Ende der Mittagspause scannen die Betreuungspersonen erneut ihren Ausweis und klicken

im Dashboard auf «Beenden», um das Angebot abzuschliessen. Anschliessend werden die iPads im Teamzimmer eingesteckt, um für den nächsten Einsatz bereit zu sein.



(a) Screenshot des Angebots «Mensa»: Die Auswertung zeigt, dass alle Kinder bereits eingecheckt sind. *(Namen zensiert)*

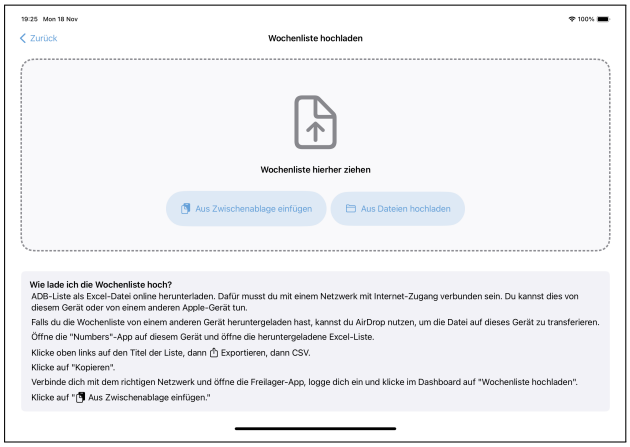


(b) Dashboard-Funktion: Anpassung der maximalen Platzanzahl. *(Namen zensiert)*

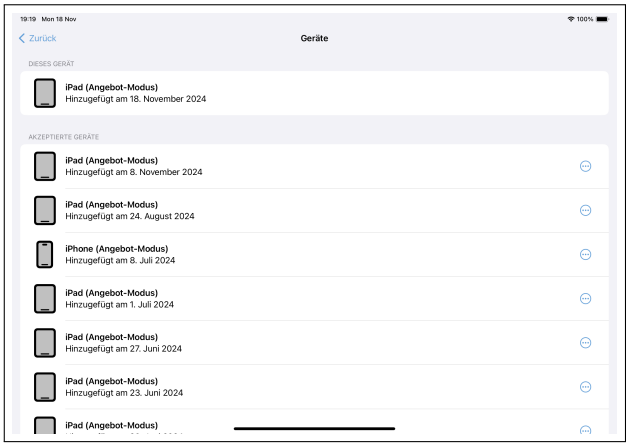
Abb. 17: Die Dashboard-Ebene im Einsatz: Auswertungen und Kapazitätsmanagement

3.5.3 Admin-Funktionen

Während ausgewählte Administratoren beim Scannen ihres Ausweises die gleichen Funktionen wie Betreuungspersonen übernehmen können, stehen ihnen im Dashboard zusätzliche Optionen zur Verfügung. Sie können Angebote erstellen, bearbeiten und löschen sowie Personen dem System hinzufügen, entweder einzeln oder als Liste. Ausserdem muss einmal pro Monat die Anwesenheitsliste aus einer internen Webseite der Stadt Zürich exportiert und von einem Administrator in das System hochgeladen werden – ein Vorgang, der ca. 5 Minuten dauert (vgl. Abb. 18a). Dabei handelt es sich eigentlich um eine Wochenliste, die jedoch für vier Wochen unverändert bleibt. Darüber hinaus gibt es im Dashboard für Administratoren die Möglichkeit, alle verbundenen Geräte zu verwalten. Das bedeutet, dass unbekannte Geräte, die sich im System einloggen, von dort entfernt werden können (vgl. Abb. 18b). Da diese Aufgaben einmalig oder nur kurz sind, reichen hier 1-2 Administratoren aus.



(a) Dashboard-Ansicht: Hochladen der Anwesenheitsliste



(b) Dashboard-Ansicht: Übersicht der ans System gekoppelten Geräte

Abb. 18: Screenshots vom Dashboard

3.6 Feedback & Verbesserungen

Das System wurde vom Verfasser durch regelmässige Besuche an der Schule Freilager beobachtet und es wurde durch Gespräche mit Beteiligten Feedback eingeholt. Die Kinder kamen von Beginn weg gut mit dem System zurecht. Da nur ausgewählte Klassen das System nutzen dürfen, ist es sehr begehrt, an der langen Schlange vor der Mensa vorbeizugehen und einfach den Ausweis zu scannen. Eine Betreuungsperson berichtete, wie sehr die Kinder eine Funktion lieben: das Konfetti. Etwa bei jedem zehnten Scan spielt die App zufällig eine Konfetti-Animation ab (vgl. Abb. 19). Es wurde auch erwähnt, dass immer wieder Kinder fragen, wann sie endlich auch das System verwenden dürften. Die Kinder hätten zudem anfänglich Schwierigkeiten gehabt, die richtige Distanz zur iPad-Kamera zu finden, damit der Barcode korrekt erfasst wird. Mittlerweile, insbesondere nach der Umstellung auf QR-Codes, sei dies jedoch kein Problem mehr. Während ein Grossteil der Rückmeldungen in direkten Gesprächen gesammelt wurde, wurde in der Pilotphase mit sechs Klassen zusätzlich eine anonyme schriftliche Umfrage durchgeführt, an der elf Betreuungspersonen der Schule Freilager teilnahmen. Die App wurde in ihrer Verständlichkeit im Durchschnitt mit 4,75 von 5 bewertet. In Bezug auf die Zuverlässigkeit erhielt die App eine Bewertung von 3,33 von 5. Hier wurde die Netzwerkverbindung als Schwachstelle kritisiert, insbesondere in Bereichen wie der Turnhalle, wo die Verbindung unzureichend sei. Dies wurde mehrfach als möglicher Verbesserungspunkt genannt. Insgesamt wurde die Frage, ob die App die Betreuungspersonen bei ihrer Arbeit unterstützt, mit einem Durchschnitt von 4,58 von 5 bewertet. Ein Kommentar lautete: «Diese App vereinfacht unsere Arbeit und gibt uns mehr Personal, um neue Angebote zu leiten. Super!!!»

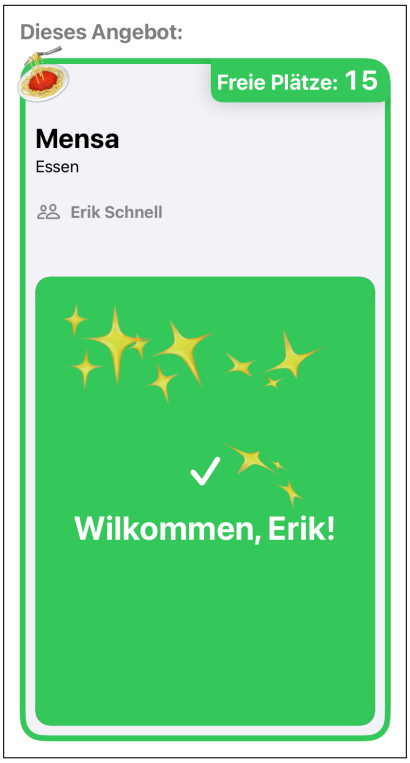


Abb. 19: Zufällige Konfetti-Animation beim Check-in

3.7 Kosten

Die Hardwarekosten sind wie folgt aufgeschlüsselt:

Hardwarekosten	**Kosten (CHF)**
iPad 9th 10.2 WiFi 64GB Space Gray (13x) [48]	3 768
Apple Mac Mini (2023) M2, 8 GB RAM, 256 GB SSD [49]	599
Evolis Badgy 100 Kartendrucker [50]	587
Router: TP-Link Archer A8 [51]	51
Verstärker: TP-Link RE505X Wi-Fi 6 WLAN Repeater AX1500 (10x) [52]	500
Leere PVC Karten (500 Stück) [53]	44
Lanyards (500 Stück) [54]	450
Gesamtkosten	**5 999**

Zusätzlich fallen jährlich schätzungsweise die folgenden laufenden Kosten an:

Laufende Kosten	**Kosten / Jahr (CHF)**
Stromkosten (Schätzung)	50
Neue PVC Karten (100 Stück)	90
Wartungskosten	500
Betriebskosten / Jahr	**640**

4 Diskussion und Ausblick

In der folgenden Diskussion wird zunächst der aktuelle Stand von ORBITS in der Schule Freilager beleuchtet. Anschliessend werden die in der Einleitung formulierten Fragestellungen anhand der gewonnenen Erfahrungen diskutiert. Abschliessend werden im Ausblick mögliche Ansätze und Ideen für die Weiterentwicklung des Systems vorgestellt.

4.1 Die App im Einsatz

ORBITS ist seit mehreren Monaten an der Schule Freilager im Einsatz, wobei es aktuell von sechs Klassen im Rahmen der Pilotphase täglich genutzt wird. Die übrigen neun Klassen sind noch nicht in das System integriert. Diese parallele Nutzung des neuen Systems und der bisherigen Methoden führt derzeit zu einem erhöhten Aufwand. Neben dem Aufstellen der iPads müssen Kinder, die noch nicht im System erfasst sind, manuell abgehakt und ihre Nutzung der Magnettafel überwacht werden. Die in der Pilotphase beteiligten Betreuungspersonen, Kinder und Administratoren äusserten jedoch überwiegend positives Feedback und sprechen sich für eine schulweite Einführung des Systems aus.

4.2 Beantwortung der Fragestellungen

4.2.1 Ist eine digitale Lösung zur Verwaltung der Betreuung überhaupt notwendig?

Eine digitale Lösung zur Verwaltung der Betreuung wird notwendig, wenn der administrative Aufwand, wie etwa das manuelle Abhaken von Anwesenheitslisten und die Überwachung der Magnettafel, erheblich ist und personelle Ressourcen bindet, wie es an der Schule Freilager der Fall ist. Die Interviews mit Schulen wie Altstetterstrasse und Allmend zeigten, dass manuelle Systeme hohen personellen Aufwand verursachen und Betreuungspersonen von ihren Kernaufgaben abhalten. Gleichzeitig wurde betont, wie wichtig es sei, die Anwesenheit und Aufenthaltsorte der Kinder im Blick zu behalten, um Sicherheit und Organisation zu gewährleisten. Eine digitale Lösung könnte in solchen Tagesschulen helfen, diese Informationen effizienter bereitzustellen. Ein positives Beispiel liefert die Schule Lachenzelg, wo eine digitale Lösung auf Basis eines QR-Code-Systems eingeführt wurde. Diese Automatisierung erhöhte die Genauigkeit und reduzierte den Aufwand für die Betreuungspersonen erheblich. Dennoch bleibt dort ein gewisser manueller Aufwand bestehen, da der Scanner weiterhin von Betreuungspersonen bedient werden muss.
Schulen wie Dachslern-Feldblumen und Hutten empfinden den administrativen Aufwand aufgrund spezifischer Gegebenheiten nicht als problematisch. In Dachslern-Feldblumen erübrigt sich eine Magnettafel, da die Kinder Kurse besuchen und keine offenen Betreuungsangebote nutzen. In der Schule Hutten macht die geringe Kinderzahl eine solche Lösung überflüssig. Diese Beispiele zeigen, dass digitale Lösungen nicht in allen Kontexten notwendig sind, sondern stark von den individuellen Gegebenheiten abhängen.
Schliesslich ist die Notwendigkeit einer digitalen Lösung nicht nur eine Frage des Bedarfs oder der Arbeitserleichterung, sondern auch der Zukunftsfähigkeit. Besonders mit einer wachsenden Anzahl von Kindern ist eine bestehende analoge Lösung kaum skalierbar. Eine digitale Lösung bietet die Möglichkeit, effizient mit grossen Datenmengen umzugehen, Prozesse zu automatisieren und gleichzeitig die Sicherheit der Kinder zu gewährleisten.

4.2.2 Wie können Betreuungspersonen durch eine digitale Lösung effektiv entlastet werden?

Die Antwort liegt in einer intuitiven und effizienten Gestaltung der Anwendung, die sich nahtlos in den Tagesablauf integriert und keine unnötige Belastung schafft. Die im Rahmen dieser Arbeit entwickelte App ist so gestaltet, dass sie im Schulalltag mit minimalem Aufwand genutzt werden kann. Die Betreuungspersonen müssen lediglich zweimal den Bildschirm berühren: einmal, um ein Angebot zu starten, und einmal, um es zu beenden.
Durch die App entfallen manuelle Aufgaben wie das Bereitstellen und Überwachen der Magnettafel sowie das Abhaken der Anwesenheitsliste, wodurch Betreuungspersonen mehr Zeit für die eigentliche Betreuung der Kinder haben. In einer Umfrage (vgl. Kap. 3.6) bestätigten die Betreuungspersonen, dass die App ihre Arbeit deutlich erleichtert und sie spürbar entlastet werden. Gleichzeitig wird gesehen, dass Betreuungspersonen, die sich im Umgang mit digitalen Geräten nicht wohlfühlen, eventuell zusätzlichen Schulungsbedarf benötigen. An der Schule Freilager hatten jedoch alle Beteiligten nach kurzer Einführung das System schnell verstanden und konnten es ohne Weiteres einsetzen.
Natürlich könnten technische Ausfälle wie Netzwerkprobleme oder Serverfehler dazu führen, dass das System vorübergehend ausfällt und Betreuungspersonen auf manuelle Prozesse zurückgreifen müssen. Eine stabile technische Infrastruktur ist daher entscheidend, um solche Ausfälle zu minimieren. In der Schule Freilager gab es bisher keine Ausfälle, jedoch wurde die Netzwerkstabilität teilweise kritisch bewertet.
Obwohl ORBITS in der Schule Freilager aktuell erst mit sechs Klassen läuft, ist absehbar, dass dessen Einführung für die gesamte Schule die Betreuungspersonen nochmals erheblich entlasten wird. Diese Erweiterung ist für die nahe Zukunft geplant (vgl. Kap. 4.3).

4.2.3 Wie wird eine digitale Lösung von den Kindern aufgenommen und genutzt?

Die digitale Lösung wurde von den Kindern in der Pilotphase gut aufgenommen, da sie durch einfache Bedienbarkeit und schrittweise Einführung schnell zur Routine wurde.
Nach einer 5-minütigen Einführung konnten die Kinder das System eigenständig nutzen und waren innerhalb weniger Wochen ohne grössere Unterstützung damit vertraut.
Das ansprechende Design der App, insbesondere das Konfetti-Feature, trug erheblich zur positiven Aufnahme des Systems bei.
Auch die schrittweise Einführung des Systems, bei der Klasse für Klasse Zugang erhielt, weckte das Interesse und die Neugier der Kinder, da sie gespannt darauf warteten, wann ihre Klasse an der Reihe ist.
Anfängliche Schwierigkeiten, den richtigen Abstand zur iPad-Kamera zu finden, um die Ausweise zu scannen, lösten sich mit zunehmender Erfahrung der Kinder auf, unterstützt durch die Umstellung auf QR-Codes, die schneller und zuverlässiger gescannt werden können.

4.2.4 Wie sollte das System gestaltet werden, um intuitiv und benutzerfreundlich zu sein?

Die Benutzerfreundlichkeit von ORBITS wird durch bekannte Technologien wie iPads und QR-Codes sowie eine einfache, intuitive Gestaltung gewährleistet.
Den meisten Kindern, Betreuungspersonen und Administratoren waren iPads bereits vertraut. Ihre benutzerfreundliche Handhabung erwies sich als Vorteil, da die grundlegenden Interaktionen mit ihnen weit verbreitet und intuitiv sind.
Ein weiterer wichtiger Aspekt ist der Einsatz von Barcodes bzw. QR-Codes. Diese sind ebenfalls ein bekanntes Konzept, bei dem die Nutzer intuitiv wissen, dass sie gescannt werden müssen. Dies senkt die Einstiegshürde und sorgt dafür, dass ORBITS von Anfang an als einfach und

zugänglich wahrgenommen wird.
Die App selbst wurde bewusst intuitiv gestaltet, gemäss den Apple Human Interface Guidelines, die bewährte Prinzipien für benutzerfreundliches Design bieten. Klare Trennung von Ebenen, Priorisierung durch Farben und Schriftarten sowie übersichtliche Container machen die Anwendung visuell ansprechend und leicht verständlich. Dabei wurde auf unnötige Interaktionen verzichtet, um die Nutzung zu vereinfachen: Kinder müssen lediglich ihre Ausweise scannen, ohne die App aktiv bedienen zu müssen, und Betreuungspersonen müssen nur einmal tippen, um ein Angebot zu aktivieren oder zu beenden. Dies verhindert ungewünschtes Herumspielen an den iPads und behält den Aufwand für die Betreuungspersonen minimal.
Trotz dieser Ansätze benötigt ORBITS weiterhin Verbesserungen. Beispielsweise kommt es gelegentlich vor, dass beim Starten eines Angebots ein schwarzer Bildschirm anstelle des Kamerafeldes angezeigt wird, was eine erneute Öffnung der App erfordert. Solche technischen Unzulänglichkeiten mindern die Benutzerfreundlichkeit und müssen kontinuierlich behoben werden, bis eine gewisse Stabilität erreicht wird.
Auch Netzwerkprobleme beeinträchtigten die Zuverlässigkeit und Benutzerfreundlichkeit des Systems, insbesondere wenn die Verbindung in bestimmten Bereichen der Schule schwächer ist. Die Betreuungspersonen der Schule Freilager gaben im Thema Netzwerkstabilität eher negative Rückmeldungen; es wird daher erforderlich sein, das Netzwerk weiter zu optimieren oder auszubauen (vgl. Kap. 4.3).

4.2.5 Welche Infrastruktur ist für die Umsetzung des Systems erforderlich?

Die notwendige Infrastruktur für das System umfasst ein separates lokales WLAN-Netzwerk, einen Mac Mini als Server, iPads als Endgeräte und QR-Codes als Identifikationsmittel, wodurch die Lösung als datenschutzfreundliche, lokal betriebene On-Premises-Architektur umgesetzt werden konnte.
Ein separates, lokales Netzwerk gewährleistete die Unabhängigkeit vom bestehenden Schulnetzwerk. Es basiert auf einem TP-Link-Router mit mehreren Mesh-Verstärkern und stellt damit eine (teilweise knapp) ausreichende WLAN-Abdeckung im gesamten Schulgelände sicher. Diese Lösung erwies sich als praktikabel, da sie eine Kommunikation aller Geräte ermöglicht, ohne das zentrale Schulnetzwerk zu belasten. Eine Option zur Verbesserung der Netzwerkstabilität wäre das Aufstellen von mehr Verstärkern.
An den verschiedenen Angeboten nutzen die Kinder iPads, um sich mit ihren Hausausweisen ein- und auszuchecken. Betreuungspersonen verwenden diese ebenfalls, um Angebote zu verwalten und Informationen über die Anwesenheit der Kinder abzurufen. Die Entscheidung für iPads erwies sich als benutzerfreundlich und vielseitig einsetzbar: Sie sind leicht zu warten und integrieren sich einwandfrei in den Schulalltag.
Die Identifikationsmittel der Kinder bestehen aus Plastikkarten mit QR-Codes, die jeweils eine individuelle ID-Nummer enthalten. Diese Lösung erwies sich als kostengünstig, zuverlässig und sicher. Im Falle eines Verlusts werden keine sensiblen Daten offengelegt, und Ersatzkarten können jederzeit nachgedruckt werden.
Der Server dient als zentrale Verwaltungseinheit des Systems. Für diesen Zweck wurde ein Mac Mini gewählt, der sowohl kompakt als auch leistungsfähig ist. Der Server speichert alle relevanten Daten und verwaltet die Anfragen der iPads. Während der Mac Mini zu Beginn durch die vielen HTTP-Anfragen überlastet war, erwies er sich nach der Umstellung auf WebSockets als effiziente und leicht wartbare Lösung.
Eine Einbindung in das bestehende Schulnetzwerk oder die Nutzung eines Cloud-Servers hätte die Infrastruktur möglicherweise vereinfacht, da keine separate Netzwerk- und Server-Hardware erforderlich wäre. Allerdings würde eine solche Lösung Datenschutzrisiken erhöhen und erfordert zudem Vertrauen in Drittanbieter.

4.2.6 Welche Programmiersprachen eignen sich für die Entwicklung des Systems?

Für die Entwicklung des Systems eigneten sich Swift und SwiftUI besonders gut aufgrund ihrer Optimierung für das Apple-Ökosystem und der damit verbundenen Performance-Vorteile. Die Nutzung von Swift im Front- und Backend ermöglichte durch die geteilte Codebasis eine effiziente und konsistente Entwicklung.
Für die Entwicklung der App wurde das Framework SwiftUI verwendet, das sich als moderne, sichere und effiziente Lösung speziell für Apple-Geräte eignet. Da die Endgeräte iPads sind, war es naheliegend, die Vorteile einer nativen Sprache zu nutzen, die eine hohe Performance sowie eine direkte Integration gerätespezifischer Funktionen, wie etwa die Nutzung der Kamera zum Scannen von QR-Codes, ermöglicht. Alternativ hätte ein plattformübergreifendes Framework wie React Native verwendet werden können, jedoch wären damit Kompromisse in Leistung und Stabilität einhergegangen, da es nicht für das Apple-Ökosystem optimiert ist.
Durch die Verwendung von Vapor als Web-Framework und Fluent als ORM-Lösung, beide basierend auf Swift, entfiel die Notwendigkeit, eine zusätzliche Programmiersprache einzusetzen. Der Vorteil, sowohl für das Backend als auch für die App Swift zu nutzen, liegt in der geteilten Codebasis. Dadurch wurde die Entwicklung effizienter, da der gleiche Code, wie etwa Datenmodelle und Schnittstellen, in beiden Anwendungen verwendet werden konnte.
Die Entscheidung, auch das Backend in Swift zu entwickeln, brachte jedoch auch Herausforderungen mit sich. Typische Server-Programmiersprachen wie Python, Node.js oder Java bieten in der Regel eine grössere Community, umfangreichere Libraries und eine bessere Dokumentation, was die Entwicklung auf dem Server wohl vereinfacht hätte. Letztlich bewährte sich Swift, da die einheitliche Sprache und geteilte Codebasis die Entwicklung effizienter und konsistenter machte.

4.2.7 Wie kann das System datenschutzkonform umgesetzt werden?

Das System wurde durch Datenminimierung, lokale und zeitlich begrenzte Datenspeicherung, anonymisierte Ausweisdaten und eingeschränkte Zugriffsrechte so gestaltet, dass es den gesetzlichen Datenschutzanforderungen entspricht. Entscheidend war, dass von Beginn an die gesetzlichen Anforderungen des Datenschutzes gemäss IDG und IDV beachtet und entsprechende technische sowie organisatorische Massnahmen integriert wurden. Indem keine überflüssigen Informationen, sondern lediglich die Anwesenheit der Kinder innerhalb der verschiedenen Angebote temporär gespeichert werden, wurde dem Prinzip der Datenminimierung entsprochen.
Ein zentraler Charakter des Systems ist die gewählte On-Premises-Lösung. Die lokale Umgebung bietet den Vorteil, dass die Kontrolle über die Daten stets bei der Schule bleibt und keine sensiblen Informationen nach aussen gelangen. Die Daten sind nur innerhalb des Schulnetzwerks zugänglich, was zusätzliche Sicherheit vor unbefugtem Zugriff bietet.
Ebenso entscheidend ist die zeitlich begrenzte Speicherung der Daten. Die Check-in-Daten der Kinder werden lediglich für den aktuellen Tag gespeichert und über Nacht gelöscht. Hierdurch wird gewährleistet, dass keine langfristige Speicherung von Bewegungsdaten erfolgt, um den gesetzlichen Vorgaben als auch dem Schutz der Privatsphäre der Kinder zu entsprechen.
Weiter wurden die eingesetzten Hausausweise derart konzipiert, dass sie keine sensiblen personenbezogenen Daten enthalten. Jeder Hausausweis ist lediglich mit einer ID-Nummer versehen, die im System einer Person zugeordnet ist. Diese ID-Nummer enthält somit keine direkten Informationen über die Person und ist für Aussenstehende nicht verwertbar.
Zusätzlich wird durch Sicherheitsmassnahmen in der App für den Schutz der Daten gesorgt. Dazu zählen die Eingabe eines Zahlencodes für den Administratorzugriff auf das Dashboard und die Begrenzung der Zugriffsrechte. Betreuungspersonen haben nur grundlegenden Zugriff, während Administratoren erweiterte Funktionen wie das Erstellen und Bearbeiten von Angeboten sowie das Verwalten von Personendaten nutzen können.

Es ist äusserst schwierig, eine vollkommen sichere Anwendung zu erstellen. Allerdings würde ein potenzielles Datenleck aufgrund der Natur des Systems nur wenige und grösstenteils uninteressante Daten preisgeben.

4.2.8 Wie kann das System flexibel gestaltet werden, um zukünftigen Anforderungen der Schule gerecht zu werden?

Das System wurde flexibel gestaltet, indem sowohl die Software als auch die Hardware so konzipiert ist, dass Anpassungen, Erweiterungen und Wechsel mit minimalem Aufwand möglich sind.
In der App selbst lassen sich durch Administratoren Personen und Angebote hinzufügen, bearbeiten oder entfernen. Neue und auch spontane Angebote können so mit minimalem Wartungsaufwand umgesetzt werden. Ebenso können neue Kinder oder Betreuungspersonen schnell und einfach hinzugefügt werden, entweder manuell oder durch das Hochladen einer vorbereiteten Liste.
Mit Blick auf die Software erweist sich das System ebenfalls flexibel. Die App wurde mithilfe eines leicht angepassten MVVM-Patterns entwickelt. Diese Architektur ermöglicht Änderungen an der Benutzeroberfläche, ohne die zugrundeliegende Logik anzupassen, und umgekehrt. Praktische Erfahrungen zeigten, dass dadurch neue Features und Änderungen effizienter und ohne Beeinträchtigung der App-Stabilität umgesetzt werden konnten.
Ein Nachteil der Verwendung von SwiftUI als Framework ist die Einschränkung auf Apple-Geräte. Ein zukünftiger Umstieg auf Android oder eine webbasierte Applikation würde eine vollständige Neuprogrammierung und damit erheblichen Entwicklungsaufwand bedeuten, was die Flexibilität des Systems in Bezug auf Plattformwechsel einschränkt. Während der Entwicklung zeigte sich SwiftUI jedoch als sehr anpassungsfähig und effizient, da Änderungen schnell und unkompliziert vorgenommen werden konnten. Innerhalb des Apple-Ökosystems ist das System daher gut gerüstet, um zukünftige Anforderungen mit geringem Aufwand zu erfüllen.
Der Server- und App-Code ist darüber hinaus klar strukturiert und nach dem DocC-Standard kommentiert. Dies stellt sicher, dass der Code auch nach einer längeren Pause oder bei Weitergabe des Projekts leicht verständlich bleibt.
Die geteilte Codebasis für Backend und App erhöht zusätzlich die Stabilität und Flexibilität der Entwicklung. Gemeinsamer Code kann zentral verwaltet werden, was zukünftige Erweiterungen erleichtert und den Anpassungsaufwand minimiert.
Schliesslich wurde auch Wert auf eine flexible Hardware gelegt. Da der Server auf Vapor läuft, das auch mit Linux-Geräten kompatibel ist, besteht die Möglichkeit, die Server-Hardware bei Bedarf zu wechseln. Da die Kommunikation über WebSockets und HTTP-Requests erfolgt, könnte die Applikation ebenfalls ausgetauscht oder aktualisiert werden, ohne dass Änderungen am Server notwendig sind. Zudem wäre ein Umstieg auf ein anderes Netzwerk oder sogar auf eine Cloud-Lösung möglich, ohne dass grösserer Anpassungsaufwand erforderlich wäre.

4.2.9 Ist die Investition in das System aus wirtschaftlicher Sicht gerechtfertigt?

Die Investition in das System ist aus wirtschaftlicher Sicht gerechtfertigt, da die dadurch eingesparte Zeit der Betreuungspersonen die anfänglichen Kosten sowie die laufenden Betriebskosten mehr als ausgleicht.
Im Vergleich zu den geringen Kosten für eine Magnettafel und einen bereits vorhandenen Laptop, um die Aufenthaltsliste abzuhaken, scheinen die Kosten von mehreren Tausend Franken für das digitale System zunächst unverhältnismässig. Jedoch berücksichtigt dieses Argument nicht, dass ORBITS nicht nur das analoge System digitalisiert, sondern darüber hinaus zahlreiche Effizienzgewinne mit sich bringt. Die Betreuungspersonen brauchen keine Magnete mehr bereitzulegen, die Magnettafel nicht mehr zu überwachen und die Kinder nicht mehr vor der

Mensa auf der Aufenthaltsliste abzuhaken. Die dadurch gewonnene Zeit kann für die eigentliche Betreuung der Kinder eingesetzt werden.
Die jährlichen Betriebskosten von ungefähr 600 Franken stellen verglichen mit den Lohnkosten der Stunden, die zuvor für administrative Aufgaben aufgewendet wurden, eine relativ geringe Summe dar. Die eingesparte Zeit durch die Entlastung der Betreuungspersonen kommt der Qualität der Betreuung direkt zugute, wodurch sich der anfängliche Investitionsbetrag und die Betriebskosten rechtfertigen lassen.

4.3 Ausblick

ORBITS ist kein abgeschlossenes Projekt – im Gegenteil: Dies war erst der Anfang. In enger Absprache mit der Schule Freilager soll das Projekt weiterentwickelt werden. Es sind Verbesserungen und die schulweite Einführung von ORBITS geplant. Zunächst wird ein Update folgen, das kleinere Mängel wie das Schwarzwerden des Kamerafelds nach dem Starten eines Angebots sowie gelegentliche Probleme beim Einpflegen der Anwesenheitsliste behebt. Zudem soll das Netzwerk auf Funklöcher und Schwachstellen geprüft und diese entsprechend beseitigt werden.
In naher Zukunft wird beabsichtigt, weitere Klassen dem System hinzugefügt werden, wobei die Server-Performance genau beobachtet wird. Solange es keine erheblichen Leistungseinbussen gibt, werden die zusätzlichen neun Klassen das System ebenfalls bald täglich nutzen. Die Magnettafel wird vollständig ersetzt, das Abhaken der Anwesenheitsliste automatisiert und, am wichtigsten, das Hauptziel von ORBITS erreicht: Die Entlastung der Betreuungspersonen.
In einem gemeinsamen Projektplan mit der Schule Freilager sind bereits weitere Features festgelegt: Mehrere Geräte sollen dasselbe Angebot übernehmen können, sodass beispielsweise zwei iPads vor der Mensa aufgestellt werden können. Kinder sollen in der App krankgemeldet werden können, damit sie in der Auswertung zum Aufenthalt beim Mittagessen nicht als fehlend, sondern korrekt als krank angezeigt werden. Kleine, arbeitserleichternde Funktionen wie der Export einer PDF-Liste der krankgemeldeten Kinder oder die direkte Einbindung des Kartendruckers, sodass der Druckauftrag direkt von einem Administrator auf dem iPad gegeben werden kann, sind ebenfalls geplant.
Eine Einführung an anderen Schulen könnte bei Interesse der Stadt Zürich eine denkbare Option werden. Die Interviews zeigten, dass die Lösung zumindest in bestimmten Schulen die Betreuungspersonen erheblich entlasten könnte. In Zusammenarbeit mit der Stadt Zürich könnte das System weiter vereinfacht werden: Die iPads könnten sich direkt mit dem schulinternen Netzwerk verbinden, sodass kein separates Netzwerk mehr erforderlich wäre. Da diese dann Zugriff auf das Internet hätten, könnte die Wochenliste automatisch heruntergeladen und eingepflegt werden. Eine Umstellung auf einen Cloud-Server, der als Backend für alle nutzenden Schulen dient, ist ebenfalls denkbar, sofern dabei der Datenschutz gewährleistet werden kann. Dies würde die Kosten für weitere Schulen erheblich reduzieren, da Hardwarekosten für Netzwerk und Server entfielen. Zusätzlich müsste eine Lösung zur Verwaltung und Aktualisierung der Server sowie der Endgeräte entwickelt werden. Beispielsweise könnten iPads mit dem Apple School Manager verbunden werden, um Updates automatisch zu installieren.
Bleibt das Projekt auf die Schule Freilager beschränkt, wird die Wartung von ORBITS durch den Verfasser möglich sein. Sobald die Entwicklung abgeschlossen ist und sich keine grösseren Änderungen ergeben, sollten im Idealfall keine weiteren Fehler auftreten. Dies kann jedoch nicht garantiert werden, weshalb gelegentlich kleinere Fehlerkorrekturen notwendig sein könnten.
Bei einer Expansion auf andere Schulen, möglicherweise in Zusammenarbeit mit der Stadt Zürich, wird die Verwaltung und Wartung von ORBITS nicht allein durch den Verfasser gewährleistet werden können. Eine Option wäre die Gründung eines Unternehmens, das Weiterentwicklung, Wartung und Support übernimmt.

Literatur

[1] Schul- und Sportdepartement Stadt Zürich. *Tagesschulen der Stadt Zürich*. https://www.stadt-zuerich.ch/ssd/de/index/volksschule/tagesschule2025.html. (Besucht am 17. 11. 2024).

[2] Kanton Zürich. *Gesetz über die Information und den Datenschutz (IDG)*. https://www.zh.ch/de/politik-staat/gesetze-beschluesse/gesetzessammlung/zhlex-ls/erlass-170_4-2007_02_12-2008_10_01-109.html. (Besucht am 17. 11. 2024).

[3] Kanton Zürich. *Verordnung über die Information und den Datenschutz (IDV)*. https://www.zh.ch/de/politik-staat/gesetze-beschluesse/gesetzessammlung/zhlex-ls/erlass-170_41-2008_05_28-2008_10_01-108.html. (Besucht am 17. 11. 2024).

[4] International Organization for Standardization (ISO). *ISO/IEC 18000-63:2021 - Information technology — Radio frequency identification for item management — Parameters for air interface communications at 860 MHz to 960 MHz Type C*. Techn. Ber. ISO, 2021. URL: https://www.iso.org/standard/78309.html (Besucht am 17. 11. 2024).

[5] Google Patents. «WO2016049847A1 - Near field communication (NFC) tag». Patent WO2016049847A1. 2016. (Besucht am 17. 11. 2024).

[6] International Organization for Standardization (ISO). *ISO/IEC 15420:2009 - Information technology — Automatic identification and data capture techniques — EAN/UPC bar code symbology specification*. Techn. Ber. ISO, 2009. URL: https://www.iso.org/standard/46143.html (Besucht am 17. 11. 2024).

[7] International Organization for Standardization (ISO). *ISO/IEC 18004:2024 - Information technology — Automatic identification and data capture techniques — QR code bar code symbology specification*. Techn. Ber. ISO, 2024. URL: https://www.iso.org/standard/83389.html (Besucht am 17. 11. 2024).

[8] Digitec Galaxus AG. *Whadda RFID Tag 5 pieces*. https://www.digitec.ch/en/s1/product/whadda-rfid-tag-5-pieces-development-boards-kits-14560445. (Besucht am 17. 11. 2024).

[9] Digitec Galaxus AG. *Cipherlab UHF RFID Reader*. https://www.digitec.ch/en/s1/product/cipherlab-uhf-rfid-reader-barcode-scanner-accessories-18083500. (Besucht am 17. 11. 2024).

[10] Schul- und Sportdepartement Stadt Zürich. *KITS für Kids - Stadt Zürich*. https://www.stadt-zuerich.ch/ssd/de/index/volksschule/kits_informatik_computer.html. (Besucht am 17. 11. 2024).

[11] Bluetooth Special Interest Group (SIG). *Bluetooth Technology Overview*. https://www.bluetooth.com/learn-about-bluetooth/tech-overview/. (Besucht am 17. 11. 2024).

[12] 3rd Generation Partnership Project (3GPP). *The Mobile Broadband Standard*. https://www.3gpp.org/. (Besucht am 17. 11. 2024).

[13] Institute of Electrical and Electronics Engineers (IEEE). *IEEE 802.11ax-2021 - High Efficiency Wireless LAN*. Techn. Ber. IEEE Standards Association, 2021. URL: https://standards.ieee.org/ieee/802.11ax/7180/ (Besucht am 17. 11. 2024).

[14] Finanzdepartement Stadt Zürich. *Organisation und Informatik - Stadt Zürich*. https://www.stadt-zuerich.ch/fd/de/index/das_departement/organisation/oiz.html. (Besucht am 17. 11. 2024).

[15] Hipp, D. Richard and Others. *SQLite, Version 3.47.0*. https://www.sqlite.org/download.html. 2024. (Besucht am 17. 11. 2024).

[16] Apple Inc. *Introducing SwiftUI: Building Your First App - WWDC19*. https://developer.apple.com/videos/play/wwdc2019/204/. (Besucht am 17. 11. 2024).

[17] Apple Inc. *SwiftUI, Version 5*. https://developer.apple.com/xcode/swiftui/. 2024. (Besucht am 17. 11. 2024).

[18] Swift Project. *Swift, Version 5.9*. https://www.swift.org/download/. 2023. (Besucht am 17. 11. 2024).

[19] Apple Inc. *Xcode, Version 15*. https://developer.apple.com/xcode/. 2024. (Besucht am 17. 11. 2024).

[20] JetBrains. *Kotlin Programming Language, Version 2.0.0*. https://kotlinlang.org/. 2024. (Besucht am 17. 11. 2024).

[21] Meta Platforms Inc. *React Native, Version 0.76*. https://reactnative.dev/. 2024. (Besucht am 17. 11. 2024).

[22] Antoine van der Lee. *MVVM: An Architectural Coding Pattern to Structure SwiftUI Views*. https://www.avanderlee.com/swiftui/mvvm-architectural-coding-pattern-to-structure-views/. (Besucht am 17. 11. 2024).

[23] Apple Inc. *SwiftUI Essentials - WWDC19*. https://developer.apple.com/videos/play/wwdc2019/216/. (Besucht am 17. 11. 2024).

[24] Apple Inc. *iOS 18*. https://www.apple.com/ios/. 2024. (Besucht am 17. 11. 2024).

[25] Apple Inc. *macOS Sequoia*. https://www.apple.com/macos/. 2024. (Besucht am 17. 11. 2024).

[26] The PHP Group. *PHP: Hypertext Preprocessor, Version 8.3.13*. https://www.php.net/. 2024. (Besucht am 17. 11. 2024).

[27] Guido van Rossum and Python Software Foundation. *Python Programming Language, Version 3.12.1*. https://www.python.org/. 2024. (Besucht am 17. 11. 2024).

[28] Yukihiro Matsumoto and contributors. *Ruby Programming Language, Version 3.3.6*. https://www.ruby-lang.org/en/. 2024. (Besucht am 17. 11. 2024).

[29] Swift Project. *Swift on Server*. https://www.swift.org/documentation/server/. 2024. (Besucht am 17. 11. 2024).

[30] R. Fielding and J. Reschke. *Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing*. RFC 7230. Internet Engineering Task Force (IETF), Juni 2014. DOI: 10.17487/RFC7230. URL: https://www.rfc-editor.org/info/rfc7230 (Besucht am 17. 11. 2024).

[31] Vapor Team. *Vapor: Web Framework for Swift, Version 4*. https://vapor.codes/. 2024. (Besucht am 17. 11. 2024).

[32] PostgreSQL Global Development Group. *PostgreSQL, Version 17*. https://www.postgresql.org/. 2024. (Besucht am 17. 11. 2024).

[33] E. Rescorla. *HTTP Over TLS*. RFC 2818. Internet Engineering Task Force (IETF), Mai 2000. DOI: 10.17487/RFC2818. URL: https://www.rfc-editor.org/info/rfc2818 (Besucht am 17. 11. 2024).

[34] I. Fette and A. Melnikov. *The WebSocket Protocol*. RFC 6455. Internet Engineering Task Force (IETF), Dez. 2011. DOI: 10.17487/RFC6455. URL: https://www.rfc-editor.org/info/rfc6455 (Besucht am 17. 11. 2024).

[35] Vapor Team. *Fluent*. https://github.com/vapor/fluent. 2024. (Besucht am 17. 11. 2024).

[36] International Organization for Standardization (ISO). *ISO/IEC 9075-1:2023 - Information Technology — Database Languages SQL — Part 1: Framework (SQL/Framework)*. Techn. Ber. ISO, 2023. URL: https://www.iso.org/standard/76583.html (Besucht am 17. 11. 2024).

[37] MongoDB Inc. *MongoDB, Version 8.0*. https://www.mongodb.com/. 2024. (Besucht am 17. 11. 2024).

[38] Martin Lasek. *Code Sharing Between iOS and Vapor*. https://www.youtube.com/watch?v=eFpXQ-VbDMo. (Besucht am 17. 11. 2024).

[39] GitHub Inc. *GitHub: Build and Ship Software on a Single Collaborative Platform*. https://github.com/. (Besucht am 17. 11. 2024).

[40] Swift Project. *Swift Package Manager*. https://www.swift.org/documentation/package-manager/. (Besucht am 17. 11. 2024).

[41] Swift Project. *DocC: Swift Documentation Compiler*. https://www.swift.org/documentation/docc/. 2024. (Besucht am 17. 11. 2024).

[42] Antoine van der Lee. *Share Swift Code Between Swift On Server Vapor and Client App*. https://www.avanderlee.com/swift/share-swift-code-swift-on-server-vapor/. (Besucht am 17. 11. 2024).

[43] Apple Inc. *Human Interface Guidelines*. https://developer.apple.com/design/human-interface-guidelines/. 2024. (Besucht am 17. 11. 2024).

[44] Google. *Designing Engaging Apps: Building for Kids*. https://developers.google.com/building-for-kids/designing-engaging-apps. (Besucht am 17. 11. 2024).

[45] Interaction Design Foundation. *Design Iteration Brings Powerful Results: So, Do It Again Designer!* https://www.interaction-design.org/literature/article/design-iteration-brings-powerful-results-so-do-it-again-designer. (Besucht am 17. 11. 2024).

[46] Jenny Bargetzi, Isabel Heusser, Oliver Camenzind. *Datenschutzbedenken: Zürcher Schule stoppt Tracking im Hort*. https://www.nzz.ch/zuerich/datenschutzbedenken-birmensdorf-stoppt-tracking-im-hort. Neue Zürcher Zeitung: 05.06.2024. (Besucht am 17. 11. 2024).

[47] Jigme Garne, Martin Huber. *Schule im Kanton Zürich wollte Kinder mit Trackern überwachen*. https://www.tagesanzeiger.ch/schule-in-birmensdorf-wollte-mit-bluetooth-armbaendern-hortkinder-tracken-379247755436. Tages-Anzeiger: 05.06.2024. (Besucht am 17. 11. 2024).

[48] Interdiscount AG. *APPLE iPad Wi-Fi 2021 9. Gen.* https://www.interdiscount.ch/de/product/apple-ipad-wi-fi-2021-9-gen-10-2-64-gb-space-grau-0002871318. Bei Aktion gekauft. (Besucht am 17. 11. 2024).

[49] Media Markt Schweiz AG. *APPLE Mac mini (2023) M2*. https://www.mediamarkt.ch/de/product/_apple-mac-mini-2023-m2-2169118.html. (Besucht am 17. 11. 2024).

[50] Digitec Galaxus AG. *Evolis Badgy 100*. https://www.digitec.ch/de/s1/product/evolis-badgy-100-300-dpi-etikettendrucker-6057660. (Besucht am 17. 11. 2024).

[51] Digitec Galaxus AG. *TP-Link Archer A8*. https://www.digitec.ch/de/s1/product/tp-link-archer-a8-router-16392714. (Besucht am 17. 11. 2024).

[52] Digitec Galaxus AG. *TP-Link RE505X Wi-Fi 6 WLAN Repeater AX1500*. https://www.digitec.ch/de/s1/product/tp-link-re505x-wi-fi-6-wlan-repeater-ax1500-1200-mbits-300-mbits-wlan-repeater-14006156. (Besucht am 17. 11. 2024).

[53] Digitec Galaxus AG. *Zebra Karten Blank 0.76mm*. https://www.digitec.ch/de/s1/product/zebra-karten-blank-076mm-lxb85x54mm-drucker-zubehoer-5808312. (Besucht am 17. 11. 2024).

[54] Ikast Etikett AG. *Unbedruckte, bedruckte Lanyards*. https://www.ikastetikett.ch/schluesselbaender_lagerware. (Besucht am 17. 11. 2024).

Anhang

Im Folgenden sind die relevanten Fragen und Antworten aus den Interviews mit Schulleitungen und Betreuungsleitungen sinngemäss transkribiert.

1 Interview Olivia Hofstter, Leitung Betreung der Schule Altstetterstrasse, 30. Mai 2024

Was machen die Kinder am Mittag?

Wir haben ein Whiteboard, auf dem die Kinder mit einem Magneten wählen können, wo sie hingehen möchten. Es gibt verschiedene Mittagsangebote. Zum Beispiel Turnhalle, Ruheraum, Tanzen, Spielzimmer oder nach draussen gehen. Die Kinder können selbstständig wählen, wo sie hingehen möchten. Entsprechend gibt es immer eine Betreuungsperson vor Ort. Diese Personen stammen aus dem gesamten Schulhaus, das heisst, es sind nicht nur Betreuungspersonen, sondern auch Klassenassistenzen und manchmal Lehrpersonen, also eine gemischte Gruppe.

Können Sie erklären, wie die Magnettafel funktioniert?

Es ist den Kindern freigestellt, wohin sie gehen. In der Turnhalle gibt es einfach eine maximale Anzahl an Kindern, die dort sein dürfen, und wir haben das nach Klassen aufgeteilt. An einem Tag sind die Grösseren in der Turnhalle und am anderen Tag die Kleineren. Ansonsten können sie sich frei bewegen. Das heisst, sie können zum Mittagessen gehen und danach ins Spielzimmer. Wenn sie dann merken, dass sie doch lieber tanzen möchten, nehmen sie ihren Magneten und wechseln ins Tanzen. Sie können sich also frei bewegen.

Wie nehmt ihr die Anwesenheit über Mittag auf?

Es sind immer zwei Klassen, die gleichzeitig essen. Die Kinder haken sich selbst auf der Liste ab, die aushängt. Eine Betreuungsperson steht daneben. Wir hätten die Liste gerne digital gemacht, aber das war zu komplex. Wir hätten das gerne umgesetzt, drucken sie aber aktuell aus. Wir haben uns auch schon überlegt, ob wir in Zukunft vielleicht gar nicht mehr kontrollieren sollten. Das wäre jedoch schwierig, weil manchmal ein Anruf von einer Mutter kommt, die fragt: «Wo ist mein Kind?» Über Mittag brauchen wir diese Kontrolle auf jeden Fall. Aber auch dann ist es schwierig, bei 150-200 betreuten Kindern genau zu sagen, wo sich ein bestimmtes Kind befindet. Vielleicht ist der Magnet am falschen Ort.

Wo werden die Absenzen erfasst?

Es gibt einen Tisch am Eingang, wo die Kinder hinkommen. Dort müssen sie sich entscheiden, wohin sie gehen möchten. Es gibt verschiedene Farben für die unterschiedlichen Angebote, und dort wird es abgehakt. Danach können die Kinder zu den Angeboten gehen, und es gibt Betreuungspersonen für jedes Angebot.

Gibt es schon digitale Lösungen dafür?

Von der Stadt aus gibt es keine. Wenn, dann müssten wir so etwas selbst im Schulhaus entwickeln. Natürlich haben wir ICT-Personen, die dafür teilweise zuständig sind, aber sie haben auch andere Aufgaben. Eine digitale Lösung wäre natürlich unser Wunsch, vor allem für das offene Betreuungsangebot. Etwas, das beispielsweise ein Tablet anzeigt: «Das sind meine Kinder, die ich betreuen sollte», vielleicht farblich markiert. Das wäre unsere Idealvorstellung, aber wir konnten das bisher nicht umsetzen.

Wie könnte eine solche Lösung aussehen, die die Magnettafel oder die Liste ersetzt?

Im offenen Betreuungsangebot hätten wir gerne eine Liste der Kinder, die sich gerade im Angebot befinden. Etwas Einfaches und Schnelles, sodass jede Betreuungsperson sofort sehen kann: «Ah, das sind meine Kinder – super!» In der Mittagsbetreuung könnte die Liste auch einfach digital sein, sodass die Kinder sich frei bewegen können. Im offenen Betreuungsangebot wäre es von Vorteil, wenn wir genau wissen, wo die Kinder sich befinden.

2 Interview mit Franziska Lehnherr, Leitung Betreuung der Schule Allmend, 19. Juni 2024

Wo halten sich die Kinder mittags auf?

Im ganzen Schulhaus. Sie müssen sich um 12 Uhr anmelden und bekommen einen Magneten mit ihrem Namen darauf, damit wir wissen, dass alle Kinder da waren. Danach können sie entscheiden, ob sie zuerst essen, spielen oder etwas anderes machen möchten. Wir haben verschiedene Gruppenräume, die thematisch eingerichtet sind. Sie können in die Bibliothek gehen, um zu lesen, oder auf das Dach, um Fussball zu spielen. Das dürfen sie frei entscheiden.

Wie funktioniert das System mit dem Magneten?

Wir haben ein grosses Whiteboard mit verschiedenen laminierten A3-Blättern. Zum Beispiel gibt es die Mensa, die 20 Plätze hat, und die Kinder können ihren Magneten dort anbringen, um sich anzumelden. Wir haben verschiedene Angebote, bei denen sich die Kinder eintragen dürfen. Wenn ein Angebot voll ist, müssen sie etwas anderes machen.

Welche Auswahl an Angeboten habt ihr?

Wir haben den Betreuungsraum, der schon vielfältig eingerichtet ist – mit einer Leseecke, einem Basteltisch und Spielen. Ausserdem gibt es spezifische Gruppenräume: ein Ruhezimmer, in dem man Pause machen und Geschichten lesen kann, die Bibliothek, die geöffnet ist, damit die Kinder Bücher anschauen und lesen können, und einen Raum mit Gesellschaftsspielen. In einem anderen Raum gibt es Materialien für Konstruktionen, wie Lego, Kappla und andere Dinge. Für die Viert- und Fünftklässler haben wir einen Chillraum, in den sie sich zurückziehen können. In einem weiteren Raum stehen zwei Tischfussballtische und eine Dartscheibe. Zu jedem Raum gibt es auf dem Magnetboard ein Blatt mit dem Namen des Raumes und der maximalen Anzahl der Plätze. Zu jedem Angebot gehört eine Betreuungsperson. Alle Räume befinden sich im gleichen Gebäude.

Wo wird die Anwesenheit der Kinder kontrolliert?

In der Mensa gibt es einen separaten Check-in, bei dem sich die Kinder anstellen und anmelden müssen. Dadurch können wir besser steuern, wie viele Kinder gleichzeitig hineingehen, damit es beim Schöpfen nicht zu voll wird. Dort haben wir eine ausgedruckte Liste mit allen Kindern, die kommen, und sie werden abgehakt. Diese Anwesenheitsliste haben wir selbst angepasst. Es ist ein relativ grosser Aufwand, das zu Beginn des Schuljahres vorzubereiten, aber danach müssen nur noch Änderungen eingetragen werden. Alle Kinder werden erfasst, und die Liste enthält verschiedene Spalten, in denen steht, wann sich welches Kind angemeldet hat. Diese Liste verwenden wir dann auch für den Zvieri, um abzuzeichnen, wer etwas bekommt. Das übernimmt jeweils eine Betreuungsperson, die das während der gesamten Mittagszeit macht.

Habt ihr schon an eine digitale Lösung gedacht?

Jein. Im ADB (Administrationssystem) sind die offene Betreuung und die Lernzeit nicht integriert, deshalb müssen wir die Liste selbst erstellen. Wir hoffen eigentlich darauf, dass es irgendwann eine einheitliche digitale Lösung von der Stadt gibt. Optimalerweise wäre diese einfach zu bedienen, um eine solche Liste zu erstellen. Wir haben überlegt, den Kindern ein Armband zu geben, mit dem sie sich einchecken könnten. Aber das ist datenschutzrechtlich heikel, und wir hatten bisher keine Zeit, diese Idee weiterzuverfolgen. Grundsätzlich würde eine digitale Lösung das Ganze erleichtern, aber es bleibt die Frage: Was macht man dann? Der Kontakt zu den Kindern ist trotzdem wichtig. Es braucht weiterhin Personen, die darauf achten, wie die Kinder auftreten, wie sie sich verhalten, und ob sie drängeln. Das müsste weiterhin gesteuert werden. Eine digitale Lösung könnte aber die Papierflut eindämmen. Trotzdem müsste aus meiner Sicht eine Person vor Ort sein, die mehr Zeit hat, sich um die Kinder zu kümmern, und nicht gleichzeitig noch die Kinder einchecken muss.

3 Interview Barbara Stocker, Leitung Betreuung der Schule Am Wasser, 4. Juni 2024

Wie ist die Magnettafel entstanden?

Ich habe die Magnettafel vor neun Jahren entwickelt. Ursprünglich hatten wir einen klassischen Hortablauf. Früher mussten die Kinder sich die Hände waschen und warten, bis alle am Tisch sassen. Wer keine Geduld hatte, machte Unfug, und es gab viel Stress. Dann haben wir beschlossen, dass wir diesen Ablauf ändern müssen, damit die Kinder nicht so lange warten müssen. Bei einer Sitzung habe ich gehört, dass es in einem anderen Schulhaus ein «Open Restaurant» gibt. Ich wusste aber nicht, wie sie das organisiert haben. Also habe ich mich mit jemandem zusammengesetzt und überlegt, wie wir die Kontrolle behalten können, ob die Kinder kommen oder nicht. Dann haben wir das Konzept geändert, sodass die Kinder nicht mehr warten müssen und selbst verantwortlich sind, indem sie ihre Magnete verschieben. Dieses System hat sich bis heute bewährt, und es ist bisher niemand auf die Idee gekommen, es zu ändern. Aber ich bin offen für alle Vorschläge, die etwas verändern könnten!

Wie funktioniert die Magnettafel bei euch?

Die Magnettafel ist eine an einem Metallgitter gespannte Tafel. Sie ist in Sektoren aufgeteilt. Im ersten Sektor werden die Magnete «geparkt». Für jedes Kind, das laut Anwesenheitsliste von den Eltern angemeldet wurde, gibt es dort einen Magneten mit seinem Namen. Diese Kinder erwarten wir im Hort. Wenn die Kinder ankommen, nehmen sie ihren Magneten, gehen entweder direkt essen oder in eines der Angebote. Dann müssen sie ihren Magneten auf dem zweiten Teil der Tafel bei ihrem Angebot platzieren. Bis zum Ende der Mittagszeit müssen alle Kinder ihren Magneten in der Küche abgegeben und etwas gegessen haben. So haben wir die Kontrolle, dass die Kinder da sind und gegessen haben.

Wie hat sich der Mittagsablauf mit der Umstellung auf Tagesschule verändert?

Mit der Tagesschule sind 20% mehr Kinder dazugekommen. Während der 90 Minuten Mittagspause in der Tagesschule können die Kinder sich nun in verschiedenen Räumen aufhalten. Die Betreuungspersonen sind in diesen Räumen zur Aufsicht eingeteilt. Für die Mittelstufe gibt es im Schulhaus zusätzliche Angebote wie die Bibliothek, die Turnhalle und offene Lernumgebungen. Die Turnhalle ist zeitlich begrenzt geöffnet. Ausserdem gibt es in der Mensa eine maximale Kinderanzahl von 24, die wir regelmässig erreichen.

Im Mittelstufenhort nutzt ihr keine Magnettafel – warum nicht?

Der Grund, warum der Mittelstufenhort keine Magnettafel verwendet, liegt wahrscheinlich daran, dass sie Angst haben, die Magnete könnten verloren gehen. Sie haben den Überblick natürlich nicht über die Striche (auf Listen). Sie versuchen immer wieder, die Kinder dazu anzuleiten, nur ihre eigenen Kreuze zu machen und nicht für andere Kinder.

4 Interview mit Karl-Heinz Viethen, Schulleiter der Schule Dachslern-Feldblumen, 23. Mai 2024

Wie werden die Absenzen der Kinder am Mittag kontrolliert?

Aktuell gibt es während der Mittagszeit keine Kontrolle der Kinder. Früher hatten wir eine Magnetwand, an der die Kinder ihre Pins für Angebote wie Fussball oder die Turnhalle anbrachten. Diese Magnetwand wurde fotografiert, und die Informationen gingen an die Betreuungsverantwortlichen. Es war ein strikt gemanagtes System, das wir inzwischen abgeschafft haben. Seit letztem Sommer haben wir ein offenes Restaurant und lassen die Kinder frei entscheiden, wohin sie gehen möchten. Es gibt keine Präsenzkontrolle, aber die Eltern werden informiert, dass die Kinder sich auf dem Schulgelände befinden. Ob ein Kind in der Turnhalle, Bibliothek oder beim Essen ist, kontrollieren wir nicht. Je mehr wir den Kindern zutrauen, je freier sie sich bewegen können und je weniger sie Magnettäfelchen aufstellen müssen, desto weniger Probleme haben wir.

Gibt es eine digitale Lösung von der Stadt Zürich?

Die Stadt Zürich bietet einige digitale Tools wie «KluS» für die Schulverwaltung und «ADB» für die Betreuung, aber diese Systeme sind oft unzureichend. Daten stimmen manchmal nicht überein, z.B. bei den Adressen eines Kindes. Mit der Einführung der Tagesschule wurde die Verwaltung vernachlässigt, und Schulen mussten improvisieren, etwa mit Excel-Tabellen. Während ADB (Administrationssystem) 2.0 für die Betreuung genutzt wird, bleibt KluS eher ein Notfalltool für Absenzen und Zeugnisse. Unsere Haltung ist: möglichst wenig Kontrolle und viel Eigenverantwortung der Kinder. Vorschläge wie Scan-Bändchen zur Überwachung lehnen wir strikt ab. Kinder sollten sich frei bewegen und unbeobachtet auch mal «verstecken» können.

Wie ist die Betreuung an der Schule gestaltet?

Unsere Betreuung umfasst normale Betreuungszeiten, Individuelle Lernzeit (ILZ) und ein Kursangebot. Für die Kurse erstellen Anbieter Steckbriefe, die über Teams mit den Klassenlehrpersonen und Schülern geteilt werden. Die Kinder wählen ihre Kurse selbst, ohne Einfluss der Eltern. Früher erhielten Eltern einen Kurskatalog, um ihre Kinder anzumelden, aber der administrative und organisatorische Aufwand war zu hoch. Heute stehen die Wünsche der Kinder im Mittelpunkt.

5 Interview mit Carolin Mlakar, Leitung Betreuung der Schule Hutten, 13. Juni 2024

Was habt ihr alles für Betreuungsangebote?

Wir haben im Schulhaus das UG zur Verfügung, das sind alte Kellerräume, die eigentlich mega cool eingerichtet sind, wo die Kinder spielen, lesen und basteln können. Es gibt auch noch so einen Tobe-Raum, der wird auch am Mittag von 12 bis 1 genutzt, das ist ein Angebot von uns. Wir haben ausserdem die Bibliothek, das offene Klassenzimmer wo auch so Bastelsachen

angeboten werden, wir haben draussen den Pausenplatz wo die Kinder sich austoben können, dann haben wir eine Musiklehrerin, die an manchen Tagen «Musik aus dem Koffer» anbietet, dann musizieren sie als kleine Gruppe unter sich.

Wie verteilen sich die Kinder über Mittag?

Die Kinder wissen, wann sie essen kommen müssen, bei 5. und 6.-Klässlern gibt's jetzt kein Zeichen mehr, bei den kleineren Kindern wird eine Glocke geläutet, wann der Wechsel stattfindet und sonst dürfen sie frei wählen. Die wuseln dann hier auf dem Pausenplatz herum und das funktioniert ziemlich gut. Und die Kontrolle wird dann beim Essen gemacht, sind alle Kinder anwesend? Das funktioniert.

Wie schaut ihr nach ob die Kinder essen kommen?

Wir haben etwa 150 bis 160 Kinder; das ist für Zürcher Verhältnisse noch klein. Wir haben eine Liste, die drucken wir jeden Tag aus und anhand von dem geht man die Kinder durch. Sind sie alle da. Je älter die Kinder werden, desto mehr ist es Eigenverantwortung, aber meistens kommen sie dann auch. Oder man fragt, manchmal erst 10 vor Ende, aber sie kommen. Eine Magnettafel ist aufgrund der überschaubaren Kinderzahl nicht nötig. Trotzdem wird die Anwesenheitsliste täglich ausgedruckt und abgehakt.

Habt ihr bereits an eine digitale Lösung gedacht?

Das würde vieles mit dem Abmelden erleichtern. Die Liste muss man jedoch trotzdem ausdrucken, da das zwei verschiedene Systeme sind. Es gibt von den Eltern Spontanbuchungen, dass sie die Kinder an Tagen, an denen sie nicht angemeldet sind, anmelden. Die Kinder müssten wir immer noch manuell abstreichen. Wir haben ja nicht 600 Kinder auf einmal, vielleicht pro Gruppe 30 Kinder, dann geht das mit der ausgedruckten Liste. Ich könnte mir eine digitale Liste vorstellen, wo man bei den Kindern ein Häkchen dranmacht, dass man weiss, das Kind ist angekommen.

6 Interview mit Daniela Steidle, Leitung Betreuung der Schule Lachenzelg, 20. Juni 2024

Hat sich die QR-Code-Lösung bewährt?

Die digitale Lösung hat sich für uns bewährt. Es geht mega schnell. Wir haben ja nur 50 Minuten Mittagspause, weil danach fängt die Lernzeit bei uns an. Und die Kids, die kommen einfach, wir können sie schnell einscannen, dann verteilen sie sich in die Räume, eine Viertelstunde später ist eigentlich der Zauber wieder vorbei. Es ist eine Verschnellerung, auch alles geworden. Es waren bereits andere Schulen da, die sich das auch angeschaut haben, es war sogar schon die OIZ da und hat sich das angeschaut, wie die funktioniert, weil das hat sich schon rumgesprochen mit der Excel-Tabelle und dem Scanner. Das ist wirklich eine tolle Sache, das vereinfacht mega viel, es geht mega schnell, die Kinder kommen rein und sie können schnell eingecheckt werden. Früher hatten wir Kärtchen, eine Liste der Kinder, die anwesend sein sollten, haben dann alle Kärtchen rausgesucht und an die Kinder verteilt. Wenn sie essen geholt haben, haben sie ihre Karten abgegeben. Das ist mit Corona gestorben. Das System funktionierte zwar, jedoch wollten wir mit Corona die Kärtchen nicht mehr berühren müssen. Jede Schülerin und jeder Schüler von unserem Schulhaus hat einen QR-Code, die haben sie aufs Handy fotografiert. Am Mittag kommen sie, zeigen den QR-Code, den scannen wir ab, dann sind sie drin. Die Excel-Tabelle hat unser Schulleiter, der auch Informatiker ist, gebaut. Und das ist eine hochkomplexe Liste.

Angefangen haben wir sie klein und dann gab es immer mehr Bedürfnisse. Und dann hat er das immer wieder so angepasst und gebaut und jetzt haben wir sie eigentlich so, wie wir sie wollen.

Was sind denn die Vorteile und die Nachteile bzw. die Schwierigkeiten, bis jetzt mit der Liste?

Ja, dass sie noch nicht ganz sauber gelaufen ist. Es hat immer noch Fehler drin, nur wenn etwas Kleines beim Namen anders geschrieben ist, z.B. ein Punkt oder zwei Punkte oder ein Abstand drin ist, dann findet sie die Liste wieder nicht. Und da sind natürlich Fehlerquellen, wenn man in die Stadt die physischen Papiere einpflegt, oder ins ADB, dann haben wir eine Liste, die wir auch noch irgendwie einpflegen müssen. Und wenn da, ja, es muss alles ganz genau stimmen, sonst geht es nicht. Seit wir diese Liste vor zwei Jahren eingeführt haben, zeigen viele Schulen grosses Interesse, weil sie eine enorme Erleichterung darstellt. Das Problem ist nur, dass, wenn die Formeln in der Excel-Tabelle nicht korrekt sind, die Berechnungen falsch ausfallen. Wir sind Sozialpädagogen, keine Informatiker, und können solche Fehler nur schwer beheben. Wir haben das Glück, dass unser Schulleiter Informatiker ist. Er hat diese Liste aufgebaut.

7 Interview mit Aimen Mokdad, Schulleiter der Schule Lachenzelg, 20. Juni 2024

Wie seid ihr auf das QR-Code-System gekommen und wie wurde es entwickelt?

Seit wir eine Tagesschule geworden sind, essen etwa 75% der 420 Schülerinnen und Schüler bei uns. Seitdem mussten wir uns überlegen, wie man den Ansturm managen kann und lange Warteschlangen vermeiden kann. Schlussendlich zahlen die Eltern Geld dafür, dass die Kinder hier essen können, darum müssen wir das Kontrollsystem haben, wer hier angemeldet ist und wer heute hier isst und wer nicht. Wir haben auch eine Meldepflicht den Eltern gegenüber, zu sagen, dass ihr Kind heute nicht zum Essen kam. Darum habe ich diese Liste gemacht. Es ist eine reine Excel-Liste, die mit einigen Formeln formatiert ist. Jeder Schüler und Schülerin bekommt eine Nummer im QR-Code-Format. Sobald die Schülerin oder der Schüler während des Mittags zum Check-in kommt, steht dort ein Hortleiter und hat den Scanner und die Liste bereit. Dann scannt er den QR-Code des Schülers oder der Schülerin, diese werden in der Excel-Liste erfasst. Nach dem Mittagessen sieht der Hortleiter nachher, wie viele Personen heute essen waren, wer nicht und wer spontan kam. Das muss auch erfasst werden und nachher gemeldet werden, dann bekommen die Eltern auch wieder eine Rechnung, da ihr Kind spontan zum Essen kam. Unsere Lösung ist rein aus Bedarf entstanden. Früher druckten wir die Wochenliste, bzw. die Monatsliste aus und die Hortleitung suchte mit dem Bleistift den Namen des Kindes in der Liste und machte einen Strich. Stell dir vor, du machst das mit 380 Kindern. Dann habe ich gedacht, das kann es nicht sein. Wir brauchten ein schnelleres System, und so ist diese Liste entstanden.