

Datenvisualisierung im Web

Verfasser: Nadrasky Dominik

Klasse: 6CAIF

Betreuer: Wenz Rene

Jahrgang: 2024/25

HTBLVA, Wien V, Spengergasse

Höhere Lehranstalt für Informatik

Ausbildungsschwerpunkt Informatik für Erwachsene

Abgabedatum: 27.06.2025

Inhalt

1.	Einleitung	4
2.	Theorie.....	4
2.1.	Was ist Datenvisualisierung?	4
2.2.	Relevanz der Datenvisualisierung im Web	5
2.2.1.	Visualisierung mit React	5
2.3.	Arten der Datenvisualisierung.....	5
2.3.1.	Statistische Diagramme.....	5
2.3.2.	Zeitreihen und Verlaufsdarstellungen	7
2.4.	Interaktive Datenvisualisierung	7
2.4.1.	Statische Visualisierungen.....	8
2.4.2.	Interaktive Visualisierungen	8
2.5.	Technologische Grundlagen.....	8
2.5.1.	API-gestützte Datenbereitstellung	8
2.5.2.	JSON als Standardformat.....	9
2.6.	Gestaltung und Wirkung von Visualisierungen.....	10
2.6.1.	Designprinzipien.....	10
2.7.	Herausforderungen bei großen Datenmengen	10
2.7.1.	Techniken zur Optimierung	11
3.	Praktische Umsetzung.....	11
3.1.	Einleitung.....	11
3.1.1.	FindMe Aufbau.....	12
3.2.	Visualisierende Komponenten im Web-Frontend	12
3.2.1.	Darstellung von Fundstücken	12
3.2.2.	Interaktive Filtermöglichkeiten	14
3.2.3.	Live-Anzeige	15
3.3.	Backend-Struktur zur Datenbereitstellung	16
3.3.1.	REST-Endpunkte für Datenzugriff	16
3.3.2.	Modellstruktur	18
3.3.3.	Filterlogik im Frontend	19

3.4.	Benutzerbezogene Visualisierung und Adminfunktionen.....	20
3.4.1.	Anzeigen & Löschen in der Adminansicht.....	20
4.	Lessons Learned	21
5.	Fachbegriffe	22
6.	Quellen und Literaturverzeichnis	24
6.1.	Abbildungsverzeichnis	24
6.2.	Abbildungsquellen	25
6.3.	Literaturverzeichnis	26

1. Einleitung

Die vorliegende projektspezifische fachwissenschaftliche Diplomarbeit handelt von Datenvisualisierung im Web im Zusammenhang der Entwicklung eines digitalen Fundbüros.

Des Weiteren sind diese Aspekte zu klären:

- Welche Technologien und Tools werden zur Datenvisualisierung im Web häufig verwendet?
- Welche gängigen Methoden und Visualisierungstechniken eignen sich besonders gut für die Darstellung von Daten im Web?
- Wie unterscheiden sich interaktive von statischen Datenvisualisierungen und welche Vorteile bietet interaktive Visualisierung?
- Wie kann die Auswahl einer Visualisierungsform die Verständlichkeit und Aussagekraft der präsentierten Daten beeinflussen?
- Welche Herausforderungen stellt die Darstellung großer Datenmengen dar, und wie können sie durch Optimierungstechniken gelöst werden?

2. Theorie

2.1. Was ist Datenvisualisierung?

Datenvisualisierung umfasst die graphische Darstellung von Informationen, um Erkenntnisse zu gewinnen und Entscheidungsprozesse zu unterstützen. Dies kann durch Diagramme, Karten oder interaktive Visuals erfolgen. Ziel ist es, Datenmuster, Trends oder Ausreißer schnell verständlich zu machen.

Neben den komplexeren Diagrammen gibt es auch einfachere und grundlegende Visualisierungsformen, die in vielen Anwendungsbereichen sehr effektiv sind. Diese einfachen Visualisierungen bieten eine schnelle Möglichkeit, Daten auf verständliche Weise darzustellen, wie herkömmliche Auflistungen.

2.2. Relevanz der Datenvisualisierung im Web

Webanwendungen stellen Informationen nicht nur in Textform da. Dank modernen Technologien, wie das React Framework, lassen sich interaktive Grafiken oder andere Visualisierungen direkt im Browser rendern. Dies erlaubt dem User, die Daten leicht zu verstehen und die Anwendung dynamisch zu nutzen. Selbst einfache Formen der Darstellung, wie Listen oder Aufzählungen, sind essenziell für Großteils des Internets.

2.2.1. Visualisierung mit React

Es lassen sich mit wenigen Code Zeilen, gut dargestellte Grafiken erstellen, die mit dem Nutzer oder Nutzerin interagieren. Wenn der Mauszeiger über bestimmte Bereiche oder Felder bewegt wird, erscheinen weitere Informationen oder Felder. Als Beispiel, lässt sich folgende interaktive Visualisierung mittels React erstellen:

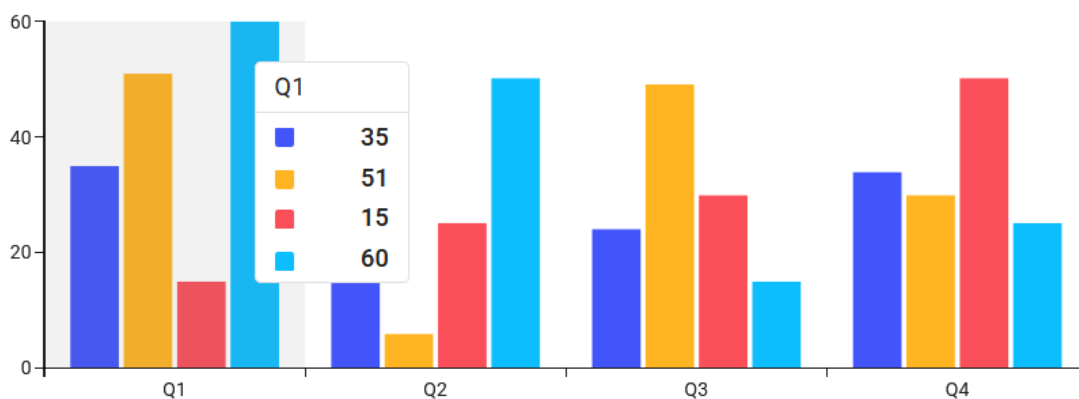


Abbildung 1: Interaktives Balkendiagramm

2.3. Arten der Datenvisualisierung

2.3.1. Statistische Diagramme

Statistische Diagramme werden gerne verwendet um so einfach und effizient wie möglich, Informationen zu vermitteln. Zu diesen gehören, als Beispiel, folgende klassische Visualisierungen wie:

- Balken- und Säulendiagramme
- Liniendiagramme
- Kreisdiagramme

Beispiel:

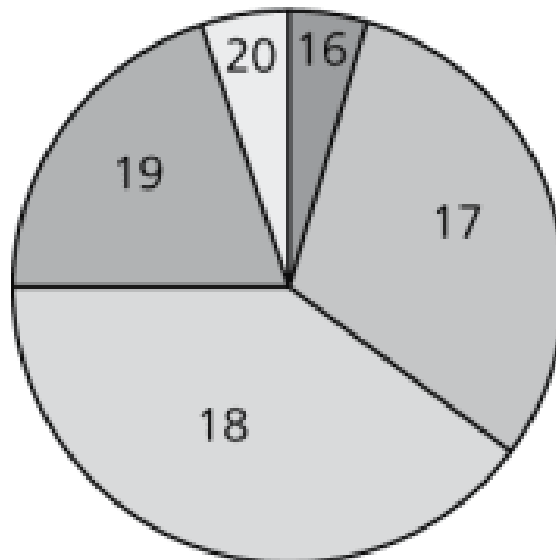


Abbildung 2: statisches Kreisdiagramm

Beispiel:

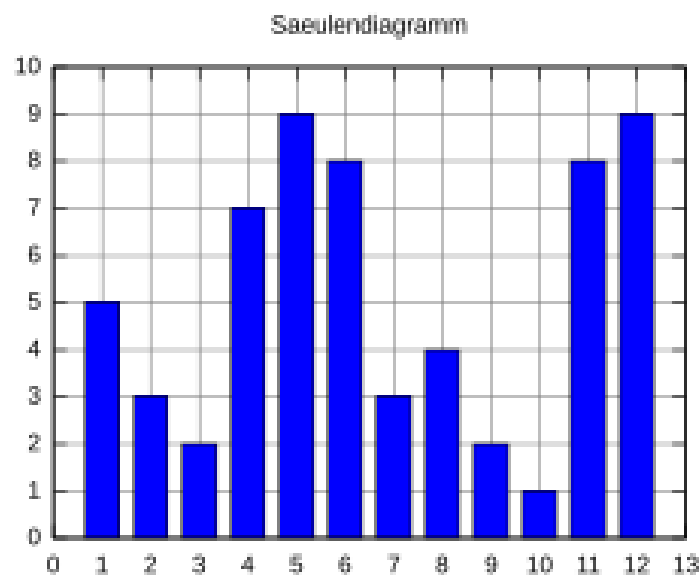


Abbildung 3: statisches Säulendiagramm

2.3.2. *Zeitreihen und Verlaufsdarstellungen*

Zu dieser Art der Datendarstellung gehören meistens nur statische Visualisierungstechniken. Es wird mit Daten im Hintergrund gearbeitet, um ein „Ergebnis“ präsentieren zu können oder auch nur bestehende Daten wie ein Verlauf dargestellt.

Als passendes Beispiel kann man das FindME Projekt betrachten. Dort wird laufend visualisiert, wie viele Gegenstände derzeit in der Datenbank als „Verloren“ eingetragen sind.

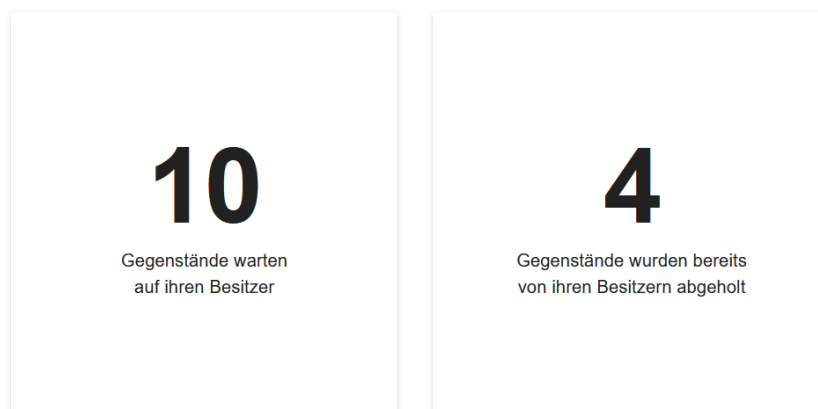


Abbildung 4: FindME Gegenstände Anzeigen

2.4. Interaktive Datenvisualisierung

Gegenüber der statischen Datenvisualisierung ist eine interaktive Visualisierung viel Benutzerfreundlicher. Der User kann meist selbst entscheiden welche Informationen er genau aus dem Diagramm oder anderen Darstellungsformen lesen möchte.

Bei der Gegenstandsauflistung, auf der FindME Webseite, kann der Nutzer oder die Nutzerin beispielsweise aktiv mit den präsentierten Daten interagieren. Er darf nach Kategorien und Fundorte filtern und so die Visualisierung nach seinen Interessen verändern.

2.4.1. *Statische Visualisierungen*

- einmalig generiert (z. B. Screenshot oder Bilddatei)
- geringere Komplexität, aber weniger flexibel
- ideal für einfache Präsentationen
- hohe Sicherheit

2.4.2. *Interaktive Visualisierungen*

- dynamisch, reagieren auf Nutzereingaben
- Zoom, Filtern, Mouseover, erweiterte Informationen
- insbesondere in Webprojekten wie **FindME** sehr vorteilhaft, da Benutzer filtern oder einzelne Datenbereiche untersuchen können

2.5. Technologische Grundlagen

React oder andere Frameworks bieten schon selbst viele Möglichkeiten für statische, aber auch interaktive Visualisierungen. Es lassen sich auch einfach, weitere Visualisierungsbibliotheken integrieren wie Chart.js, D3.js, Recharts, Google Charts oder viele weitere. Mit diesen ist es möglich, Daten nach seinen Wünschen zu präsentieren.

2.5.1. *API-gestützte Datenbereitstellung*

Das Backend stellt Daten, als Beispiel, über [/api/statistics/byMonth](#) bereit. Diese Endpunkte aggregieren Daten so, dass sie sofort von Visualisierungsbibliotheken genutzt werden können.

Die Daten sind meistens als JSON-Objekt formatiert und werden so übergeben.

2.5.2. *JSON als Standardformat*

In der Regel wird mit dem JSON-Format gearbeitet, was mit Schlüssel-Wert-Paaren aufgebaut ist.

JSON steht für JavaScript Object Notation und ist heute das Standardformat für den Datenaustausch zwischen Server und Client, besonders im Web. Es ist leichtgewichtig, menschenlesbar, gut strukturiert und sehr gut mit JavaScript-basierten Frameworks (wie React) kompatibel.

Im Kontext der Datenvisualisierung ist JSON das verbindende Element zwischen dem Backend, das Daten bereitstellt, und dem Frontend, dass diese Daten visuell aufbereitet. Egal, ob es sich um einfache numerische Werte oder komplexe Objektstrukturen handelt. JSON stellt eine Struktur zur Verfügung, um alle Arten von Informationen zu übertragen.

Beispiel JSON-Objekt:

```
[  
{ "monat": "Januar", "anzahl": 12 },  
{ "monat": "Februar", "anzahl": 9 },  
{ "monat": "März", "anzahl": 17 },  
]
```

2.6. Gestaltung und Wirkung von Visualisierungen

Die Wahl der richtigen Darstellungsmöglichkeit ist entscheidend für viele Aspekte, wie Aussagekraft oder Verständlichkeit. Dabei kann man grundlegende Regeln einhalten, um die am besten passende Visualisierung zu nutzen. Wenn man Verteilungen zeigen will, nutzt man oft Kreisdiagramme, aber auch Balkendiagramme erzielen häufig dieselbe Aussagekraft. Doch wenn man, als Beispiel, einen Zeitverlauf präsentieren möchte, sollte man ein Liniendiagramm verwenden, ein Balkendiagramm würde hier nicht passend sein.

2.6.1. *Designprinzipien*

Wichtig bei der Gestaltung der Visualisierungen ist jeder Teil der Darstellung, doch meist erzielen die einfachsten Präsentationsmittel, die beste Wirkung. Doch selbst die falsche Farbe könnte einen Nutzer oder einer Nutzerin daran hindern, Informationen wie gewünscht oder auch schnell zu verstehen.

Prinzipien:

- Farbcodierung, Lesbarkeit, Kontraste
- Tooltips für Details
- Reduktion auf das Wesentliche

2.7. Herausforderungen bei großen Datenmengen

Bei großen Datenmengen oder Datenpunkten entstehen schnell Performance- und Darstellungsprobleme. Übliche Problematiken, die auftreten werden:

- Browserleistung sinkt
- Unübersichtliche Darstellungen („Datenüberflutung“)
- Lange Ladezeiten bei API-Aufrufen

2.7.1. *Techniken zur Optimierung*

Auf fast jeder Webseite, die mit größeren Datenmengen arbeitet, werden Techniken zur Optimierung eingesetzt, um das Nutzererlebnis performant zu halten.

Selbst einfachere Techniken können schon vieles bewirken. Bei den meisten Darstellungen lassen sich „Verbesserungen“ oder „Vereinfachungen“ vornehmen, wie standardmäßig nur wenige Daten zu zeigen und den Nutzer oder Nutzerin selbst interaktiv, die gewollten Daten auszuwählen oder zu filtern. Auch kann man oft Informationen summieren und zusammenfassen, was die Leistung erheblich verbessern kann.

Komplexere Techniken wie „Lazy Loading“ oder serverseitige Filterung sind stark optimierende Möglichkeiten. Man kann „harte“ Beschränkungen direkt am Server einstellen, so dass, als Beispiel, nur mehr Daten von diesem Jahr gezeigt werden.

Als „Lazy Loading“ bezeichnet man die moderne Technik, nur die Daten zu laden die auch im Sichtbereich sind.

3. Praktische Umsetzung

3.1. Einleitung

Die Webanwendung *FindME* dient der Erfassung, Verwaltung und Suche von Fundgegenständen in einer Bildungseinrichtung. User können Fundstücke einsehen und zurückfordern, Administratoren pflegen die Daten.

In diesem Projekt sind auch mehrfach erwähnte Visualisierungstechniken genutzt worden, um dem Nutzer oder der Nutzerin der Anwendung, die nötigen Informationen bestmöglich zu vermitteln.

3.1.1. FindMe Aufbau

Backend:

Das Backend von FindME wurde mit ASP.NET entwickelt und klassische Entwicklungsstrukturen eingehalten. Es wurde eine REST-API mit Endpunkten für Items, Kategorien, Räume und auch Benutzerverwaltung implementiert und eine Datenbankverbindung mittels Entity Framework Core verwendet.

Frontend:

Das Frontend ist mittels React.js mit MUI (Material UI) geschrieben worden. Durch Komponentenbasierte UI, haben wir all unsere nötigen Visualisierungen erstellen können. Dazu gehören einfache Auflistungen, aber auch Eingabemasken, Filterfelder oder Dialoge.

Format:

Als Schnittstelle zwischen dem Frontend und dem Backend wurde JSON genutzt. Beide Bereiche können mit diesem Format arbeiten und so optimal die Daten anzeigen, verändern oder zu speichern.

3.2. Visualisierende Komponenten im Web-Frontend

In der Anwendung FindME wurden keine klassischen Diagramme oder Charts verwendet, um Informationen zu liefern. Stattdessen nutzen wir interaktive Listen, Filter oder Item Cards zur Darstellung.

3.2.1. Darstellung von Fundstücken

Die Liste der Fundstücke wurde mit dynamischen Item-Kacheln (Item-Cards) aufgebaut. Diese enthalten jeweils immer ein Bild, Titel, Kategorie, Fundort und eventuell eine Beschreibung. Bei einem Klick auf eines der Gegenstand-Kacheln, öffnet sich ein Detaildialog (vergrößerte Detailansicht) mit mehr Informationen.

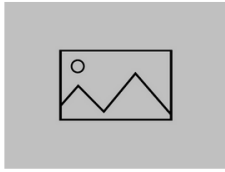
Nach Name suchen



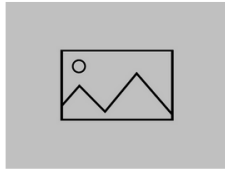
Kategorie filtern



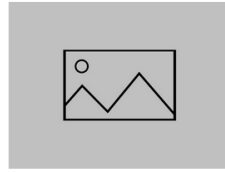
Raum filtern



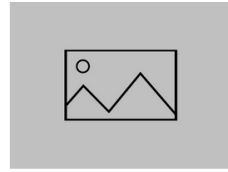
Laptop Netzteil/Ladekabel
C2.09



Grüne Jacke
DE.11W



Iphone
C4.10



Laptop
B1.08L

Abbildung 5: FindME Fundstücke (Testdaten)

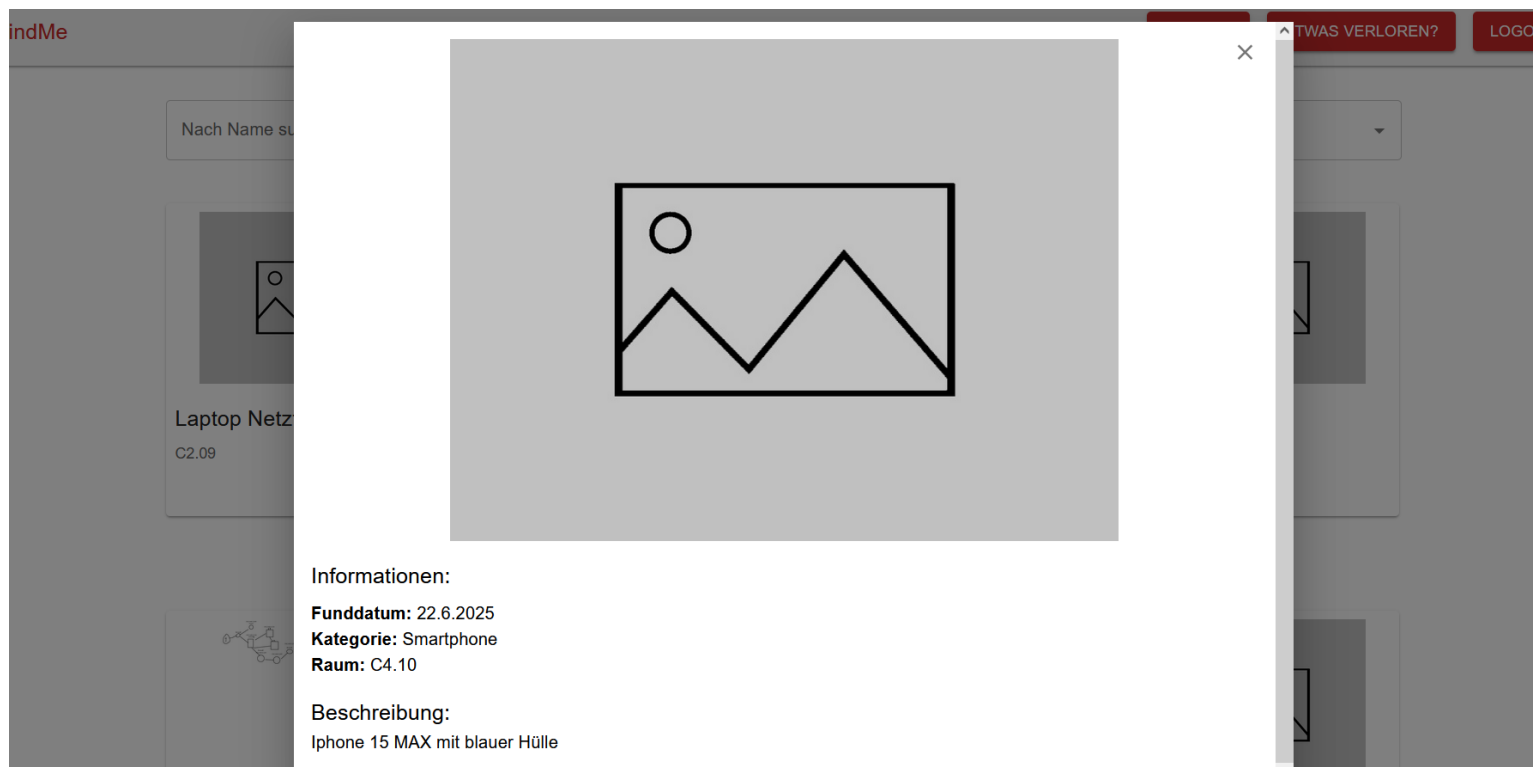


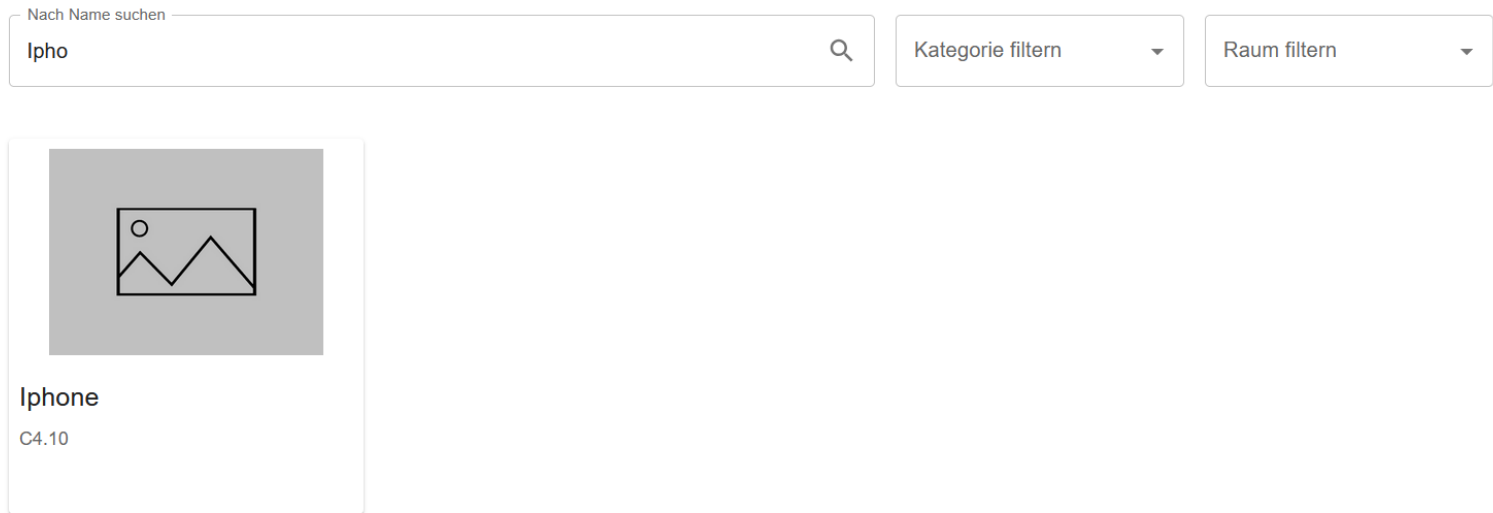
Abbildung 6: FindME Detailansicht (Testdaten)

(Da dies Testdatensätze sind, gibt es keine Bilder, was im Betrieb nicht möglich ist)

3.2.2. Interaktive Filtermöglichkeiten

Interaktive Filtermöglichkeiten erlauben es dem User die Fundliste nach seinen Interessen anzupassen. Dies wurde mit Client-seitiger Filterung umgesetzt.

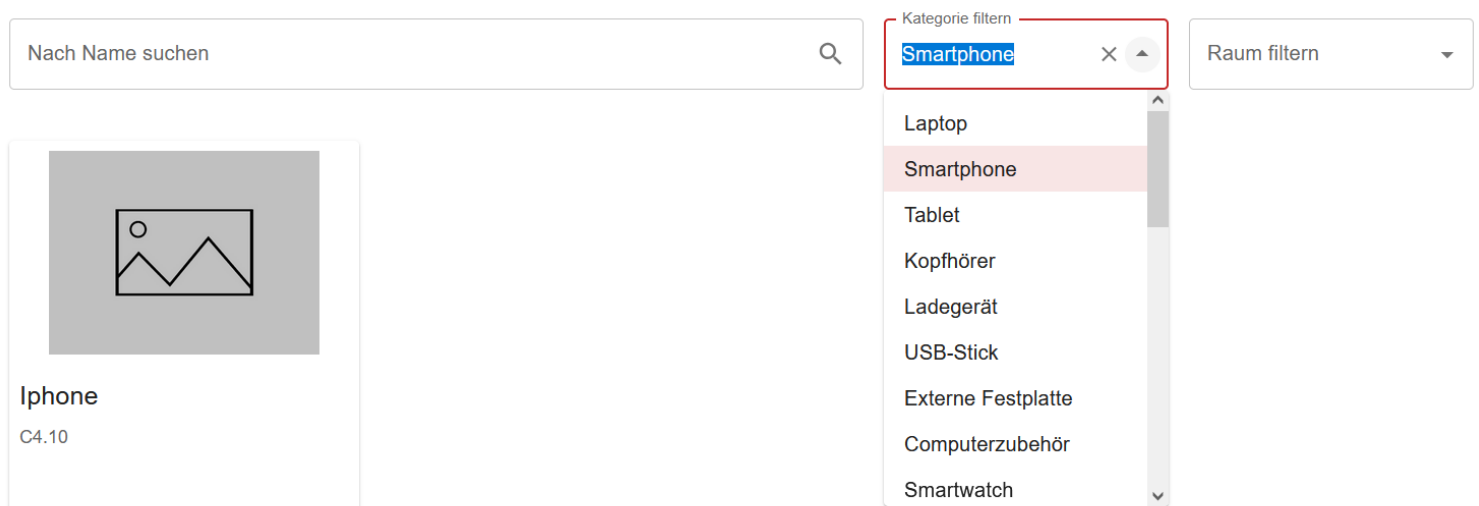
Suche nach Name: Textfeld, das Client-seitig die Liste filtert



The screenshot shows the FindME search interface. At the top, there is a search bar labeled 'Nach Name suchen' containing the text 'Ipho'. To the right of the search bar are two filter buttons: 'Kategorie filtern' and 'Raum filtern'. Below the search bar, a card for an 'Iphone' is displayed with a placeholder image and the text 'C4.10'.

Abbildung 7: FindME Namefilter

Filter nach Raum oder Kategorie: Dropdown-Listen (Autocomplete)



The screenshot shows the FindME search interface with the 'Kategorie filtern' dropdown menu open. The search bar still contains 'Ipho'. The dropdown menu lists various categories: Laptop, Smartphone (highlighted), Tablet, Kopfhörer, Ladegerät, USB-Stick, Externe Festplatte, Computerzubehör, and Smartwatch. The 'Raum filtern' button is also visible.

Abbildung 8: FindME Kategoriefilter (mit Dropdown)

3.2.3. *Live-Anzeige*

Die Daten werden im Frontend nach einer Eingabe sofort neu dargestellt und somit laufend aktuell gehalten. Dies ist ein perfektes Beispiel für eine einfache Form von interaktiver Datenvisualisierung und zeigt ebenfalls, wie effektiv solche Darstellungen sind.

Jeder der die FindME Seite besucht, wird in den ersten Sekunden erkennen können wie viele Gegenstände derzeit im Fundbüro liegen und erlaubt es schnell Informationen auf einem Blick zu vermitteln.

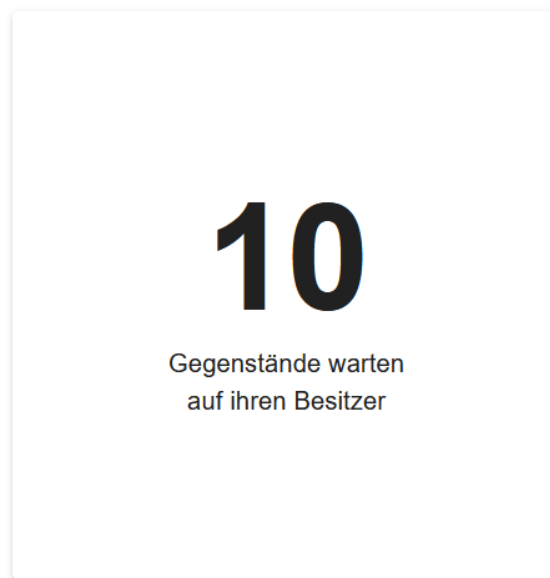


Abbildung 9: FindME Fundstücke Card

3.3. Backend-Struktur zur Datenbereitstellung

3.3.1. REST-Endpunkte für Datenzugriff

Um die Daten, die man im Frontend sieht, auch von dem Backend zu bekommen sind REST-Endpunkte nötig gewesen. Folgende Endpunkte wurden erstellt, damit von dem jeweiligen Endpunkt die korrekten Daten im JSON-Format an das Frontend geleitet werden. Folgende Endpunkte wurden implementiert und übermitteln bei einem Aufruf die JSON-Daten:

- **GET /api/item/mapped** – Alle Fundstücke mit zugehöriger Kategorie & Raum
- **GET /api/category** – Kategorien
- **GET /api/location** – Räume

```

▼ 0:
  name: "Laptop"
▼ 1:
  name: "Smartphone"
▼ 2:
  name: "Tablet"
▼ 3:
  name: "Kopfhörer"
▼ 4:
  name: "Ladegerät"
▼ 5:
  name: "USB-Stick"
▼ 6:
  name: "Externe Festplatte"
▼ 7:
  name: "Computerzubehör"
▼ 8:
  name: "Smartwatch"
▼ 9:
  name: "Taschenrechner"
▼ 10:
  name: "Elektronik"
▼ 11:
  name: "Laborausrüstung"

```

Abbildung 10: API Category Endpunkt JSON


```
▼ 0:
  name: "A1.03"
▼ 1:
  name: "A1.05"
▼ 2:
  name: "A1.06"
▼ 3:
  name: "A1.14"
▼ 4:
  name: "A1.15"
▼ 5:
  name: "A1.16"
▼ 6:
  name: "A1.17"
▼ 7:
  name: "A1.23"
▼ 8:
  name: "A2.04"
▼ 9:
  name: "A2.05"
▼ 10:
  name: "A2.06"
▼ 11:
  name: "A2.13"
▼ 12:
  name: "A2.14"
▼ 13:
  name: "A2.20"
▼ 14:
  name: "A3.04"
▼ 15:
  name: "A3.05"
▼ 16:
  name: "A3.12"
```

Abbildung 11: API Location Endpunkt JSON

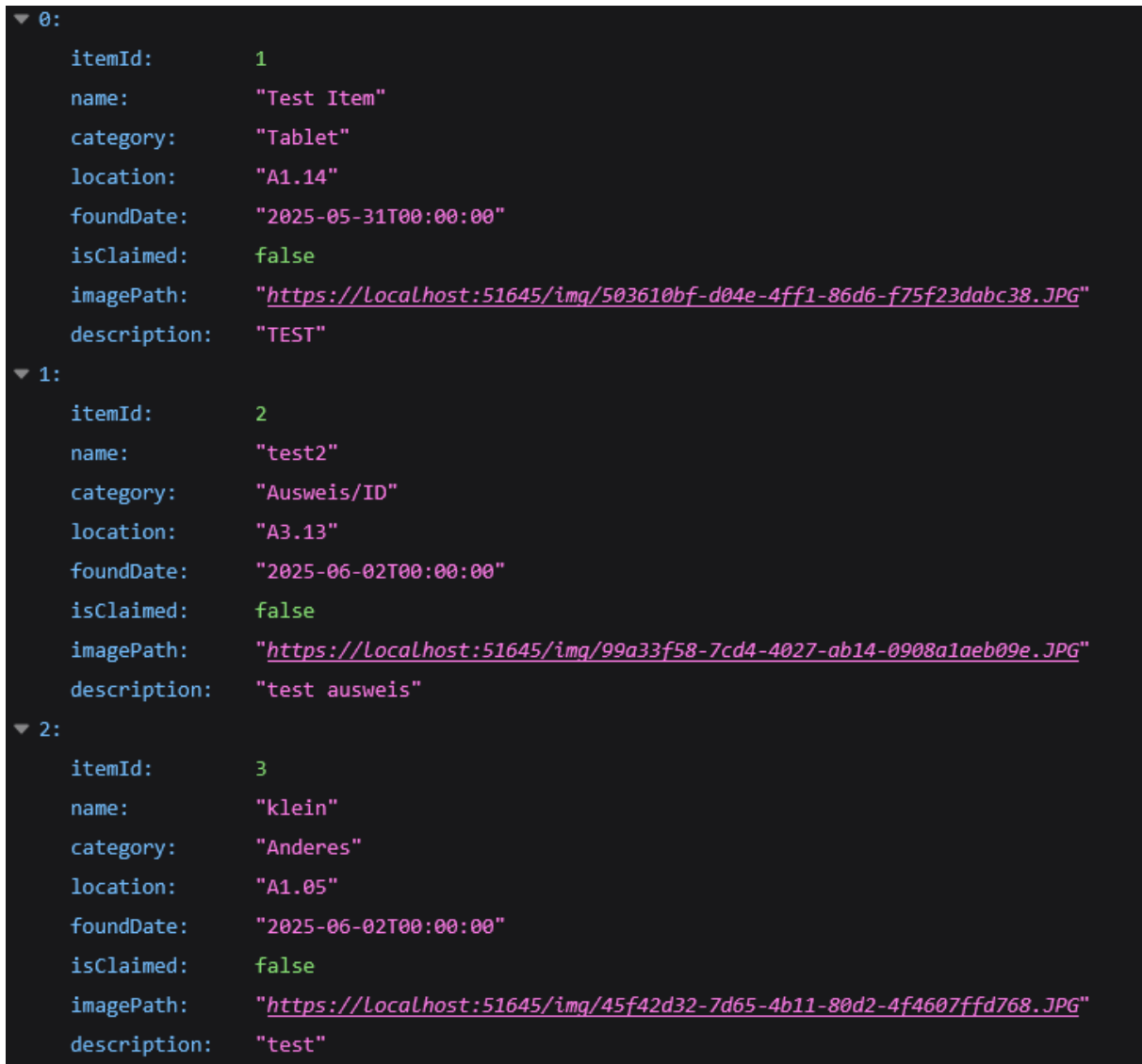


Abbildung 12: API Item/mapped Endpunkt JSON

3.3.2. Modellstruktur

Die Modellstruktur wurde einfach, kompakt und effizient gestaltet. Es gibt insgesamt vier Datenmodelle. Dazu gehören Category, Location und Item, wobei hier eine 1:n Beziehung besteht, damit Items den Kategorien und Fundorten zugeordnet werden können.

Es existiert ebenfalls eine Accountstruktur, wobei zwei statische Account implementiert wurden. Diese dient aber rein zur leichten Verwaltung und Authentifizierung des „Admin“ und „Verwalter“ Account.

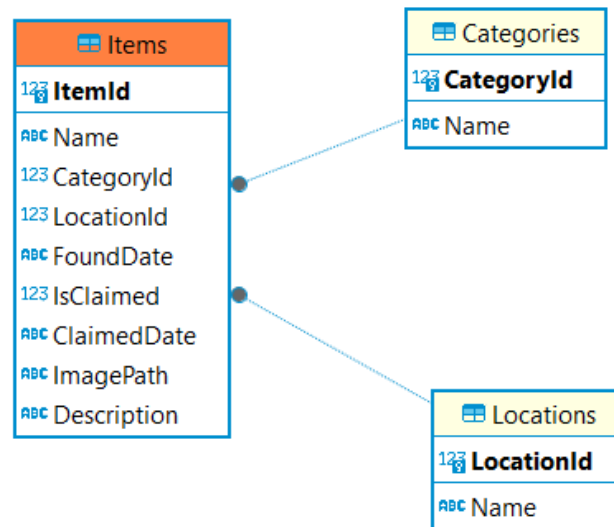


Abbildung 13: FindME DB ER-Diagramm

Die Account Klasse wird nicht als eigene Tabelle gespeichert, da die Daten statisch sind.

3.3.3. Filterlogik im Frontend

Die Filterlogik kam häufig zum Einsatz und wurde in mehreren React Komponenten implementiert. Als Beispiel, nenne ich hier die schon Erwähnte Fundstück-Liste, die sich durch folgende Eingabemöglichkeiten steuern lässt:

- Ein Texteingabefeld für die Namenssuche
- Ein Dropdown zur Auswahl der Kategorie
- Ein Dropdown zur Auswahl des Raums

Die Filterung erfolgt im Frontend durch folgende Logik:

```
54
55 ✓ const filteredItems = items.filter((item) => {
56     const matchesName = item.name.toLowerCase().includes(searchName.toLowerCase());
57     const matchesCategory = !selectedCategory || item.category === selectedCategory;
58     const matchesLocation = !selectedLocation || item.location === selectedLocation;
59     return matchesName && matchesCategory && matchesLocation;
60 });
```

Abbildung 14: Frontend Filterlogik

Diese drei Bedingungen (Z. 56, 57, 58) sorgen dafür, dass die Ergebnisliste reaktiv angepasst wird, sobald ein Filter gesetzt oder verändert wird.

Die Filter wirken dabei kombinatorisch, das heißt alle aktiven Filterbedingungen müssen gleichzeitig erfüllt werden, um das jeweilige Ergebnis anzuzeigen.

3.4. Benutzerbezogene Visualisierung und Adminfunktionen

Die statischen Accounts Admin und Verwalter haben mehr Funktionen und Zugriffe als ein üblicher User. Der Admin darf in den Adminbereich, wo auch exklusive Visualisierungen und Funktionen zur Verfügung stehen.

3.4.1. *Anzeigen & Löschen in der Adminansicht*

Hier ist klar die Verwaltung im Vordergrund, jedoch wurden auch hier Filterfunktionen zur erleichterten Verwaltung implementiert. Wie bei der herkömmlichen Gegenstand-Suche, sind die Filter- und Suchfelder reaktiv. Als Beispiel, kann man mit wenig Aufwand Kategorien suchen und ebenfalls mit einem Klick löschen. Dabei werden Dialogfenster aufgerufen, die das Löschen bestätigen und sicherer macht.

Kategorien

Suchen

Sm

Smartphone



Smartwatch



[Weniger anzeigen](#)

Abbildung 15: Adminbereich Kategorien Verwaltung

Bei einem Klick auf das Lösch-Icon, öffnet sich ein Dialogfenster zur Bestätigung, dieses prüft auch noch ob Items existieren, die diesen Wert noch in der Datenbank verwenden. Falls dies der Fall ist, wird der Löschvorgang nicht zugelassen.

4. Lessons Learned

Ich habe sehr viele neue Dinge lernen können und dass nicht nur in Technischer Hinsicht. Das allgemeine Entwickeln von einem Backend, API's und Datenbanken bzw. wie diese miteinander arbeiten oder in Verbindung stehen, verstehe ich jetzt weitreichend. Vor dem Projekt habe ich rückblickend nicht wirklich verstanden, was es heißt, wenn das Frontend mit dem Backend interagiert oder was genau eigentlich mit den Endpunkten passiert. Nun bin ich in der Lage, eine moderne Backendstruktur aufzubauen, die nötigen Methoden, Klassen, aber auch grundlegende Backend Konfigurationen zu implementieren.

Auch habe ich sehr viel lernen können, wie man unter „realen“ Bedingungen in einem Team arbeitet und ein Projekt umsetzt. Die Schwierigkeiten einer guten Planung, Umsetzung und Absprache, zeigten mir was wirklich wichtig bei so einer Gruppenarbeit ist und worauf in der Zukunft mehr bzw. weniger Wert legen sollte.

5. Fachbegriffe

Frontend: Der Teil einer Webanwendung, der für den Benutzer sichtbar und zugänglich ist. Es umfasst das Design und die Benutzeroberfläche (UI), die direkt mit dem Nutzer interagiert.

Backend: Der Teil einer Webanwendung, der für die Logik, die Datenverarbeitung und die Kommunikation mit der Datenbank zuständig ist. Das Backend stellt APIs zur Verfügung und verarbeitet die Anfragen vom Frontend.

API: Eine Schnittstelle, die es verschiedenen Softwarekomponenten ermöglicht, miteinander zu kommunizieren. APIs stellen Funktionen bereit, die von externen Programmen (wie dem Frontend) genutzt werden können, um Daten zu erhalten oder zu senden.

REST-API: Eine spezifische Art von API, die auf den Prinzipien von REST (Representational State Transfer) basiert und Webanwendungen eine einfache und flexible Möglichkeit bietet, mit anderen Anwendungen zu kommunizieren.

Datenmodell: Eine Struktur, die beschreibt, wie Daten gespeichert und organisiert werden.

Mappen: Der Prozess des Zuordnens oder Abgleichens von Daten aus verschiedenen Quellen oder Formaten.

React.js: Eine JavaScript-Bibliothek zur Erstellung von Benutzeroberflächen, die hauptsächlich für das Frontend verwendet wird. React ermöglicht es, wiederverwendbare UI-Komponenten zu erstellen, die mit dem Zustand und den Daten einer Anwendung interagieren.

Material UI (MUI): Ein React-Komponenten-Framework, das eine Sammlung vorgefertigter UI-Komponenten bietet, die den Material Design-Richtlinien von Google folgen. Es erleichtert das Erstellen von Benutzeroberflächen, indem es Design-Komponenten für Buttons, Dialoge, Formulare usw. bereitstellt.

Lazy Loading: Eine Technik zur Optimierung der Ladegeschwindigkeit von Webseiten, bei der nur die Daten geladen werden, die sich im sichtbaren Bereich der Webseite befinden. Alle anderen Daten werden erst nach Bedarf nachgeladen.

Entity Framework Core: Ein Open-Source ORM (Object-Relational Mapping) Framework für .NET, das die Interaktion mit einer relationalen Datenbank vereinfacht.

Item-Karten (Item Cards): Eine Art der Darstellung von Fundstücken im Frontend, die relevante Informationen wie Bild, Titel, Kategorie und Fundort enthalten. Diese Karten sind interaktiv und ermöglichen es dem Benutzer, mehr Details durch Klicken zu sehen.

Datenüberflutung (Data Overload): Ein Zustand, bei dem eine große Menge an Daten zu einer Überforderung des Nutzers führen kann, was die Interpretation der Daten erschwert.

6. Quellen und Literaturverzeichnis

6.1. Abbildungsverzeichnis

Abbildung 1: Interaktives Balkendiagramm.....	5
Abbildung 2: statisches Kreisdiagramm.....	6
Abbildung 3: statisches Säulendiagramm	6
Abbildung 4: FindME Gegenstände Anzeigen	7
Abbildung 5: FindME Fundstücke (Testdaten)	13
Abbildung 6: FindME Detailansicht (Testdaten)	13
Abbildung 7: FindME Namefilter	14
Abbildung 8: FindME Kategoriefilter (mit Dropdown)	14
Abbildung 9: FindME Fundstücke Card	15
Abbildung 10: API Category Endpunkt JSON	16
Abbildung 11: API Location Endpunkt JSON	17
Abbildung 12: API Item/mapped Endpunkt JSON	18
Abbildung 13: FindME DB ER-Diagramm	19
Abbildung 14: Frontend Filterlogik	20
Abbildung 15: Adminbereich Kategorien Verwaltung.....	21

6.2. Abbildungsquellen

Abbildung 1: <https://mui.com/x/react-charts/>

Abbildung 2: <https://learnattack.de/schuelerlexikon/mathematik/statistische-diagramme>

Abbildung 3: <https://de.wikipedia.org/wiki/S%C3%A4ulendiagramm>

Abbildung 4: Von FindME Projekt

Abbildung 5: Von FindME Projekt

Abbildung 6: Von FindME Projekt

Abbildung 7: Von FindME Projekt

Abbildung 8: Von FindME Projekt

Abbildung 9: Von FindME Projekt

Abbildung 10: Von FindME Projekt

Abbildung 11: Von FindME Projekt

Abbildung 12: Von FindME Projekt

Abbildung 13: Von FindME Projekt

Abbildung 14: Von FindME Projekt

Abbildung 15: Von FindME Projekt

6.3. Literaturverzeichnis

Datenvisualisierung (17.06.2025) Von <https://www.ibm.com/de-de/topics/data-visualization>

Designprinzipien. (kein Datum). Von <https://www.ibm.com/de-de/topics/rest-api> abgerufen

MUI (2025 08 03) Von <https://mui.com/x/react-charts/>

IT, R. (10. 11 2022). Frontend vs. Backend Development: Breaking Down the Difference. Von <https://medium.datadriveninvestor.com/frontend-vs-backend-development-breaking-down-the-difference-f9b1c09ae8ff> abgerufen