

# Prototyping Projektdokumentation

---

Name: Kiyan Rassouli

E-Mail: rassokiy@students.zhaw.ch

URL der deployten Anwendung: <https://fangmonitor.netlify.app>

# Inhaltsverzeichnis

<b>1. Einleitung .....</b>	<b>3</b>
<b>2. Datenmodell .....</b>	<b>3</b>
<b>3. Beschreibung der Anwendung .....</b>	<b>5</b>
<b>3.1. Startseite .....</b>	<b>5</b>
<b>3.1.1 Fangstatistik nach Seen .....</b>	<b>6</b>
3.1.1.1 Datenstruktur von /fischfangstatistik .....	6
3.1.1.2 Statistiken für den ausgewählten See .....	7
3.1.1.3 Statistik-Typen für den ausgewählten See im ausgewählten Jahr .....	7
3.1.1.4 Details für den gewünschten Statistik-Typ .....	8
<b>3.2. Fischlexikon .....</b>	<b>9</b>
3.2.1. Lexikon nach Fischkategorie .....	10
<b>3.3. Neuen Fang eintragen .....</b>	<b>11</b>
3.3.1. See Auswählen .....	12
3.3.2. Fischart auswählen .....	12
3.3.3. Länge eingeben .....	13
3.3.4. Gewicht eingeben .....	13
3.3.5. Datum & Uhrzeit erfassen .....	13
3.3.6. Standort auswählen .....	14
3.3.7. Weitere Details .....	15
3.3.8. Fang Speichern .....	15
3.3.9. Fehlende Formularelemente .....	15
3.3.10. Gespeicherte Daten .....	15
<b>4. Erweiterungen .....</b>	<b>16</b>
<b>4.1. Umfangreicher Einsatz des MongoDB Aggregation Frameworks zur Datenstrukturierung .....</b>	<b>16</b>
4.1.1.1 async function getSingleFangstatistikGroup(jahrParam, seeldParam, typParam) .....	16
<b>4.2. Effiziente Nutzung von distinct() für dynamische Elemente und Navigation .....</b>	<b>17</b>
<b>4.3. Spezifische Datenzugriffslogik für das Single-Document Fischlexikon .....</b>	<b>17</b>
<b>4.4. Integration der Browser Geolocation API zur Standortermittlung .....</b>	<b>17</b>
<b>4.5. Anbindung der externen Open-Meteo Wetter API .....</b>	<b>18</b>
<b>4.6. Hierarische Navigation für Statistiken .....</b>	<b>18</b>
<b>4.7. Dynamische Optionen für Dropdowns aus Datenbank .....</b>	<b>18</b>
<b>4.8. Robuste Fehlerbehandlung .....</b>	<b>18</b>

## 1. Einleitung

In der Schweiz ist die Erfassung der Fischereistatistiken kantonal geregelt. In vielen Kantonen erfolgt dies noch über Stift und Papier. Das Ziel dieser Anwendung soll sein, die Erfassung zu vereinfachen und vereinheitlichen. Ein weiterer Bestandteil der App ist die interaktive Gewässerübersicht: Für jeden unterstützten See können Nutzerinnen und Nutzer die Entwicklung der Fangmengen nach Fischart und Jahr einsehen. Zudem zeigt die Anwendung ein Fischlexikon mit den häufigsten Fischarten der Schweiz, so dass Einsteiger sich direkt einen Überblick verschaffen können, bevor sie ans Wasser gehen.

Zudem bietet FangMonitor eine datenbasierte Webapplikation zur Analyse, Visualisierung und Erfassung von Fischfängen in Schweizer Seen. Die Anwendung vereint amtliche Fangstatistiken mit Echtzeit-Wetterinformationen und ermöglicht eine ortsbezogene Dokumentation individueller Fänge. Ziel ist eine genauere Datenbasis für Fischereistatistiken zu erhalten, wo Ort, Zeit und aktuelle Wetterentwicklung dokumentiert sind, zugunsten der Fischerei und der Fischereibewirtschaftung. Die Daten

FangMonitor richtet sich an Freizeitfischerinnen und Fischer, Behörden und Umweltinteressierte, die Fischfangdaten systematisch auswerten, dokumentieren oder vergleichen möchten wissenschaftlich fundiert, visuell aufbereitet und regional fokussiert. Weg vom Kantönligeist hin zu einer einheitlichen Datengundlage.

## 2. Datenmodell



Die Relationen in diesem NoSQL-Kontext werden nicht durch Datenbank-Constraints erzwungen, sondern durch die Anwendungslogik in db.js und den Server-Dateien hergestellt, wenn Daten abgefragt und kombiniert werden.

### 1. Erfasster Fang (catches) und Fischlexikon (fischlexikon):

Die Anwendung verknüpft einen Eintrag in «catches» mit dem «fischlexikon» über das Feld `catches.fischart_name`. Dieser Name entspricht dem `name`-Feld eines Fischeintrags im Fischlexikon. Dies ermöglicht es der Anwendung, Detailinformationen zu einem gefangen Fisch aus dem Lexikon zu ziehen oder Vorschläge im Eingabeformular zu machen (via `getAllFischartenNamen` in db.js).

### 2. Erfasster Fang (catches) und dem entsprechenden See (aus Fangstatistik):

Die Anwendung verwendet das Feld `catches.see_id`, um einen Fang einem See zuzuordnen. Die Liste der gültigen Seen für die Eingabe wird dynamisch aus den eindeutigen `see_id`-Werten der Fangstatistik-Collection generiert (via `getUniqueLakeds` in db.js). Dies stellt sicher, dass Fänge für Seen erfasst werden, für die auch historische Vergleichsdaten existieren könnten.

### 3. Statistikanzeige (basierend auf fangstatistik) und Fischlexikon (fischlexikon):

Bei der Anzeige der aggregierten Statistiken (z.B. in der Detailansicht einer Statistikgruppe) enthält die fangstatistik-Collection die fischart\_id. Die Anwendung nutzt diese fischart\_id (die dem name-Attribut im Lexikon entspricht), um den vollständigen, benutzerfreundlichen Namen des Fisches aus dem Fischlexikon darzustellen. Dies geschieht typischerweise über eine Mapping-Tabelle oder direkte Logik im Frontend-Code (Zugriff via db.js auf getAllFischartenNamen in den Svelte-Komponenten).

### 3. Beschreibung der Anwendung

#### 3.1. Startseite

Route: routes/

## Willkommen zum FangMonitor

Alles rund um die Fischerei in der Schweiz

### Navigation

[Fischfangstatistiken Anzeigen](#)

[Fischlexikon Durchstöbern](#)

[Neuen Fang Eintragen](#)

### Über diese Anwendung

Sehen Sie detaillierte Statistiken über Fischfänge in den 5 grössten Schweizer Seen ein. Helfen Sie der Fischbewirtschaftung, indem Sie direkt selbst Fänge erfassen und ein umfangreiches Fischlexikon konsultieren.

© 2025 FangMonitor. Ein Studentenprojekt.

Die Startseite ist schlicht und man kann direkt zu den Grundfunktionen der Applikation navigieren.

Fischfangstatistiken anzeigen führt zur Übersicht der erfassten Seen.

Fischlexikon durchstöbern führt zu der Übersicht der verschiedenen Kategorien, wo man schliesslich nach Fischartenkategorie ein entsprechendes Lexikon konsultieren kann.

Neuen Fang eintragen führt zum Formular für das Eintragen von Fängen.

Datei: routes/+page.svelte

### 3.1 Fangstatistik nach Seen

Route: /fischfangstatistik

#### Fangstatistik nach Seen

Wählen Sie einen See aus, um die Jahresstatistiken anzuzeigen:



[Zurück zur Startseite](#)

Nach Auswahl von «Fangstatistiken anzeigen» kann man hier das gewünschte Gewässer auswählen, wobei man zu fischfangstatistik/[see\_id] gelangt. Im Moment sind die fünf grössten Schweizer Seen erfasst. Durch die Auswahl des gewünschten Gewässers wird man weiter zu den Jahresübersichten seit 2013 geleitet. Wenn man mit der Maus über den gewünschten See fährt, wird dieser dunkelblau eingefärbt (siehe hier Bodensee). Mit Klick auf «Zurück zur Startseite» gelangt man entsprechend auf die Startseite.

Dateien:

- lib/server/db.js → via getUniqueLakelds
- src/routes/fischfangstatistik/+page.server.js
- src/routes/fischfangstatistik/+page.svelte

#### 3.1.1. Datenstruktur von /fischfangstatistik

Die Daten für die Fischfangstatistiken sind in der MongoDB-Datenbank FangMonitor gespeichert. Die Collection in dieser Datenbank heisst fangstatistik. Die Daten sind wie folgt strukturiert:

The screenshot shows a MongoDB interface with a sidebar containing collections: FangMonitor, fangstatistik, fischlexikon, see, and userfang. The fangstatistik collection is selected. To its right, a document is displayed with the following fields and values:  
\_id: ObjectId('6820923d16bbf11bab9a494d')  
see\_id: "bodensee"  
jahr: 2013  
fischart\_id: "egli"  
typ: "Total"  
menge\_kg: 39497

Für jeden See gibt es pro Typ, Fisch und Jahr ein Dokument.

- Typ bezeichnet hierbei, ob es sich bei der Statistik um die Angel- oder Berufsfischerei handelt, oder ob beide Statistiken zusammengefasst wurden.
- see\_id: Im String-Format, jeder See teilt eine ID
- fischart\_id: Gleich wie bei see\_id. Die historischen Daten haben Fangstatistiken für die Fischarten- bzw. Übergruppen der Fische Egli, Hecht, Felchen, Cypriniden, Forellen, Saiblinge und Restliche Arten erfasst. Dies hat den Hintergrund, dass diese Fischarten für die Fischerei am interessantesten sind und Fangstatistiken der jeweiligen Kantonalen Jagd- und Fischereiverwaltungen nur die Erfassung bestimmter Arten explizit verlangen.

Die Daten sind dort noch sehr roh und nicht im Format, welches für die Applikation angewendet wird. Aus diesem Grund werden diese mit Mongo-DB Aggregationen in db.js angepasst. Dies wird im Kapitel Erweiterungen genauer erläutert.

Die Daten für die Fischfangstatistik stammen von:

<https://www.fischereistatistik.ch/de/statistics?tt=0&dt=0&at=0&st=0&dp=0&ar=CH&wt=0&th=10&un=0&in=0&yr%5Bfrom%5D=2000&yr%5Bto%5D=2023&sp=100102>

### 3.1.2. Statistiken für den ausgewählten See

Route: routes/fischfangstatistik/[see\_id]

**Statistiken für Bodensee**

Wählen Sie ein Jahr aus, um die Statistik-Typen anzuzeigen:

[Statistiken für das Jahr 2023](#)

[Statistiken für das Jahr 2022](#)

[Statistiken für das Jahr 2021](#)

[Statistiken für das Jahr 2020](#)

[Statistiken für das Jahr 2019](#)

[Statistiken für das Jahr 2018](#)

[Statistiken für das Jahr 2017](#)

[Statistiken für das Jahr 2016](#)

[Statistiken für das Jahr 2015](#)

[Statistiken für das Jahr 2014](#)

[Statistiken für das Jahr 2013](#)

[Zurück zur Seenübersicht](#)

[Zurück zur Startseite](#)

In diesem Beispiel wurde in /fischfangstatistik der Bodensee ausgewählt. Hier kann man für den ausgewählten See jeweils noch das gewünschte Jahr auswählen. Die Anzeige reicht von 2013 bis 2023, da vor 2013 keine einheitlichen Daten verfügbar waren und (Stand 11.5.2025) noch keine Daten für das Jahr 2024 verfügbar waren. Durch «hovern» über das gewünschte Jahr, verändert sich die Farbe entsprechend. Außerdem kann man unten über die Links zurück zur Seenübersicht oder direkt zurück auf die Startseite navigieren.

Dateien:

- lib/server/db.js → via getUniqueYearsForSee
- routes/fischfangstatistik/[see\_id]/+page.server.js
- routes/fischfangstatistik/[see\_id]/+page.svelte

### 3.1.3. Statistik-Typen für den ausgewählten See im ausgewählten Jahr

Route: routes/fischfangstatistik/[see\_id]/[jahr]

**Statistik-Typen für Bodensee im Jahr 2023**

Wählen Sie einen Statistik-Typ aus, um die Details anzuzeigen:

[Angelfischerei](#)

[Berufsfischerei](#)

[Total](#)

[Zurück zur Jahresauswahl für Bodensee](#)

[Zurück zur Seenübersicht](#)

[Zurück zur Startseite](#)

Auf dieser Seite werden für den ausgewählten See und das ausgewählte Jahr die zur Verfügung stehenden, historischen Fangstatistiken angezeigt. Wie bei den Seiten zuvor, wird auch hier via «hover» die gewünschte Route dunkel eingefärbt. Außerdem gelangt man über die Links «Zurück zur Jahresauswahl für [see\_id]», «Zurück zur Seenübersicht» und «Zurück zur Startseite» auf die gewünschte Seite zurück. Mit Klick auf den gewünschten Statistik-Typ gelangt man auf die detaillierte Statistik des gewünschten Sees, für das gewünschte Jahr und den gewünschten Typ.

- lib/server/db.js → via getUniqueTypesForSeeYear
- routes/fischfangstatistik/[see\_id]/[jahr]/+page.server.js
- routes/fischfangstatistik/[see\_id]/[jahr]/ +page.svelte

### 3.1.4. Details für den gewünschten Statistik-Typ

Route: routes/fischfangstatistik/details/[see\_id]/[jahr]/[typ]

**Details für: Bodensee - 2023 - Angelfischerei**

Allgemeine Informationen der Gruppe

Jahr: 2023  
See: Bodensee  
Typ: Angelfischerei

Aufstellung der gefangenen Fischarten

Fischart	Menge
Egli	4219.00 kg
Felchen	1099.00 kg
Hecht	6929.00 kg
Cypriniden (div.)	916.00 kg
Forellen	1160.00 kg
Saiblinge	302.00 kg
Restliche Arten	1424.00 kg

[Zurück zur Typ-Auswahl für Bodensee, 2023](#)  
[Zurück zur Seenübersicht](#)  
[Zurück zur Startseite](#)

Die Seite zeigt die Übersicht der gewünschten Statistik. Die Statistik wird auf Jahresebene geführt und ausgewertet. Die Menge bezeichnet also die Summe aller Fänge, gemessen in Kilogramm für die verfügbaren Fischarten. Die Funktionalität beschränkt sich auf die Ansicht der jeweiligen Statistik, wobei auch hier jeweils die Zurücklinks implementiert wurden.

Dateien:

- lib/server/db.js → via getSingleFangstatistikGroup
- routes/fischfangstatistik/details/[see\_id]/[jahr]/[typ]/+page.server.js
- routes/fischfangstatistik/details/[see\_id]/[jahr]/[typ]/+page.svelte

### 3.2. Fischlexikon

Route: routes/fischlexikon

## Fischlexikon

Entdecken Sie Informationen zu verschiedenen Fischkategorien.

### Kategorien

#### Raubfische

Fische, die sich vorwiegend von anderen Fischen und Tieren ernähren.

#### Salmoniden

Fische aus der Familie der Lachsfische, oft in kühleren Gewässern.

#### Cypriniden

Karpfenfische, eine artenreiche Familie von Süßwasserfischen.

#### Andere

Weitere einheimische Fischarten.

#### Neozoen

Gebietsfremde Arten, die durch menschlichen Einfluss eingeführt wurden.

Wählen Sie eine Kategorie aus, um mehr über die darin enthaltenen Fischarten zu erfahren. Diese Sektion wird laufend erweitert.

[Zurück zur Startseite](#)

Hier wurde die Gestaltung von der Fischfangstatistik abgegrenzt, indem grün als Grundfarbe dient. Die Fischkategorien sind hier nach Kategorien aufgeschlüsselt. Insgesamt sind 30 Fischarten erfasst. Diese werden wiederum in fünf Kategorien unterteilt. Mit hovern über die gewünschte Kategorie, wird diese dunkel eingefärbt und unterstrichen. Mit Klick auf die Kategorie wird weiter navigiert auf /fischlexikon/[category].

Auch hier kann direkt auf die Startseite navigieren.

Dateien:

- routes/fischlexikon/+page.svelte
- static/images

### 3.2.1. Lexikon nach Fischkategorie

Route: routes/fischlexikon/[category]

## Lexikon: Raubfische

Anzahl Fische in dieser Kategorie: 5



### Zander (*Sander lucioperca*)

**Habitat:** Flüsse und Seen

**Max. Grösse:** 100 cm

Ein beliebter Raubfisch, der in vielen Seen und Flüssen der Schweiz vorkommt. Er hat einen markanten Körperbau und ist für seine Jagdgeschicklichkeit bekannt.



### Egli (*Perca fluviatilis*)

**Habitat:** Flüsse und Seen

**Max. Grösse:** 50 cm

Der Egli ist ein weit verbreiteter Raubfisch in der Schweiz. Er lebt in klaren, kühlen Gewässern und ist aufgrund seiner Größe und seiner Kampfkraft ein beliebtes Ziel für Angler.



### Wels (*Silurus glanis*)

**Habitat:** Flüsse, tiefe Seen

**Max. Grösse:** 300 cm

Der Wels ist ein riesiger Raubfisch und kann in vielen Schweizer Gewässern gefunden werden. Er ist bekannt für seine imposante Größe und seine Jagdfähigkeiten.



### Hecht (*Esox lucius*)

**Habitat:** Seen und Flüsse

**Max. Grösse:** 150 cm

Der Hecht ist ein klassischer Raubfisch, der in vielen Gewässern der Schweiz vorkommt. Er jagt mit schnellen Bewegungen und scharfen Zähnen.



### Aal (*Anguilla anguilla*)

**Habitat:** Flüsse, Seen

**Max. Grösse:** 120 cm

Der Aal ist ein scheuer Raubfisch, der in den Gewässern der Schweiz heimisch ist. Er ist bekannt für seine schlangenartige Form und seine Beweglichkeit im Wasser.

[Zurück zur Kategorieübersicht](#)  
[Zurück zur Startseite](#)

Hier gibt es pro Fischart der entsprechenden Kategorie je einen Steckbrief. Als Titel wird der Name, mit der wissenschaftlichen Bezeichnung in Klammern angezeigt. Ausserdem wird ein neutrales Bild der jeweiligen Fischart abgebildet. Im Steckbrief abgebildet sind das Habitat, die maximale Grösse und eine Kurzbeschreibung.

Auch hier kann man in der Fussnote der Seite entweder auf die Kategorienübersicht oder auf die Startseite navigieren.

Die Daten stammen aus dem Buch «Fische der Schweiz» von BirdLife Schweiz in Zusammenarbeit mit dem Schweizerischen Fischereiverband SFV. Die Daten selbst wurden dann via ChatGPT in JSON-Format gebracht und in der Collection «fischlexikon» in der Datenbank FangMonitor in MongoDB gespeichert. Die Datenstruktur dort sieht

wie folgt aus:

```
_id: ObjectId('683b21651c37079d9da855eb')
  Raubfische: Array (5)
    0: Object
      name: "Zander"
      scientific_name: "Sander lucioperca"
      description: "Ein beliebter Raubfisch, der in vielen Seen und Flüssen der Schweiz vo..."
      image_url: "/images/zander.jpg"
      category: "Raubfisch"
      habitat: "Flüsse und Seen"
      max_size_cm: 100
    1: Object
    2: Object
    3: Object
    4: Object
    Salmoniden: Array (6)
    Cypriniden: Array (10)
    Andere: Array (4)
    Neozonen: Array (5)
```

Der Zugriff auf MongoDB folgte via db.js und im Frontend wiederum mit page.server.js. Relevant für diese Seite in db.js ist die Funktion getFischForCategory.

Dateien:

- lib/server/db.js → via getFischForCategory
- routes/fischlexikon/[category]/+page.server.js
- routes/fischlexikon/[category]/+page.svelte

### 3.3. Neuen Fang eintragen

Route : routes/fang-eintragen

## Neuen Fang eintragen

**Fisch & Fangdetails**

See:  
-- See auswählen --

Fischart:

Länge (cm):

Gewicht (kg):

**Datum & Uhrzeit**

Datum:  
03.06.2025

Uhrzeit:  
17:36

**Standort**

Ort (Beschreibung):  
z.B. Nähe Steg, Schilfbereich Westufer

**Aktuellen Standort verwenden**

**Weitere Details (Optional)**

Köder:

  

Bemerkungen:

**Fang speichern**

[Zurück zur Startseite](#)

Hier kann der User einen Fang mit Hilfe eines Formulars eintragen.

Mit einem Klick auf das Dropdownmenü «-- See auswählen --» wird eine Liste von Seen angezeigt, welche in der Datenbank vorhanden sind.

Dateien:

- lib/server/db.js → getUniqueLakelds und getAllFischartenNamen, schreibt Dateien via addCatch
- routes/fang-eintragen/+page.server.js
- routes/fang-eintragen/+page.svelte

### 3.3.1. See Auswählen

## Neuen Fang eintragen

**Fisch & Fangdetails**

See:

✓ -- See auswählen --
  
 Bodensee
   
 Genfersee
   
 Neuenburgersee
   
 Vierwaldstättersee
   
 Zürichsee

### 3.3.2. Fischart auswählen

Das Freetextfeld ermöglicht es, aus dem Dropdownmenü eine Fischart einzugeben, oder auch explizit zu suchen. Die Fischarten stammen aus der MongoDB-Collection «fischlexikon» und sind alphabetisch geordnet.

Aal
   
 Agone
   
 Alet
   
 Bachforelle
   
 Barbe
   
 Brachsmen

Mit Suche:

Fischart:

A screenshot of a dropdown menu titled "Fischart". The input field contains the letter "H". Below it, a list of fish species is shown: Hecht, Felchen, Trüsche, and Äsche. The first item, "Hecht", is highlighted with a blue border.

### 3.3.3. Länge eingeben

Bei Länge eingeben kann der User die Menge in cm eintragen.

Länge (cm):

A screenshot of a numeric input field. The value "60" is displayed. To the right of the input field is a small control panel with a left arrow and a right arrow, used for incrementing or decrementing the value by 0.1 cm.

Ein Klick auf den Pfeil ermöglicht es in 0.1 cm Schritten, die Länge anzupassen

Länge (cm):

A screenshot of a numeric input field. The value "60,1" is displayed. To the right of the input field is a small control panel with a left arrow and a right arrow, used for incrementing or decrementing the value by 0.01 cm.

### 3.3.4. Gewicht eingeben

Hier kann der User das Gewicht in kg erfassen

Gewicht (kg):

A screenshot of a numeric input field. The value "2" is displayed. To the right of the input field is a small control panel with a left arrow and a right arrow, used for incrementing or decrementing the value by 0.01 kg.

und in 10g, also 0.01kg Schritten adjustieren.

Gewicht (kg):

A screenshot of a numeric input field. The value "1,99" is displayed. To the right of the input field is a small control panel with a left arrow and a right arrow, used for incrementing or decrementing the value by 0.001 kg.

### 3.3.5. Datum & Uhrzeit erfassen

Hier werden Datum- und Uhrzeit angezeigt, bzw. eingegeben.

#### Datum & Uhrzeit

A screenshot of a form section titled "Datum & Uhrzeit". It contains two input fields: "Datum:" with the value "03.06.2025" and "Uhrzeit:" with the value "17:36". Both fields have small control panels to the right for adjusting the values.

Durch klick auf das Datums- bzw. das Uhrzeitfeld kann beides jeweils angepasst werden:

## Datum & Uhrzeit

Datum:

03.06.2025



Juni 2025 ▾

M	D	M	D	F	S	S
26	27	28	29	30	31	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	1	2	3	4	5	6

Löschen      Heute

Uhrzeit:

17:55



17	55
18	56
19	57
20	58
21	59
22	00
23	01

dort verwenden

### 3.3.6. Standort auswählen

Der Standort kann im Freitextfeld optional näher Beschrieben werden.

#### Standort

Ort (Beschreibung):

z.B. Nähe Steg, Schilfbereich Westufer

Aktuellen Standort verwenden

Durch Klick auf «Aktuellen Standort verwenden», wird der aktuelle Standort ermittelt und beim Absenden des Formulars mitgesendet. Die Längen- und Breitengrade sind auf sechs Nachkommastellen genau. Dies bedeutet, orientiert am Äquator, eine Genauigkeit von ca. 11 cm. Aufgrund der Erdabplattung strebt dieser Wert gegen Null, je näher man sich den Polen nähert. In der Schweiz beträgt diese Genauigkeit 7-8 cm.

#### Standort

Ort (Beschreibung):

z.B. Nähe Steg, Schilfbereich Westufer

Aktuellen Standort verwenden

Lat: 47.650867, Lon: 9.131341 (wird mitgesendet)

### 3.3.7. Weitere Details

#### Weitere Details (Optional)

Köder:

Weisser Wobbler von Rapala, 12cm, langsam sinkend

Bemerkungen:

Der Fisch hat im Freiwasser gebissen.

Hier kann man jeweils optional noch weitere Details wie Köderwahl und allgemeine Bemerkungen verfassen.

### 3.3.8. Fang Speichern

Durch Klick auf Fang speichern, erscheint die Bestätigung ganz oben unter dem Titel.

Fang speichern ←

## Neuen Fang eintragen

Fang erfolgreich eingetragen!

### 3.3.9. Fehlende Formularelemente

Fehlen Pflichtfelder im Formular, so werden diese jeweils wie folgt angezeigt:

- Fehlender See

#### Fisch & Fangdetails

See:

-- See auswählen --

Objekt von der Liste auswählen

- Fehlende Fischart:

#### Fisch & Fangdetails

See:

Bodensee

Feld ausfüllen

Fischart:

I

?

- Fehlender Standort:

## Neuen Fang eintragen

Standort (Latitude/Longitude) ist erforderlich. Bitte Standort ermitteln.

Pflichtfelder sind also Standort, See und Fischart, die anderen Daten sind keine Pflichtfelder.

### 3.3.10. Gespeicherte Daten

Auf die Details der Datenerfassung soll im Kapitel «Erweiterungen» eingegangen werden. Hier eine grobe Erklärung. Es werden natürlich sämtliche Userdaten erfasst und in der MongoDB-Collection «catches» gespeichert. Die Daten enthalten jedoch nebst den Userdaten auch noch Wetterdaten, die im Hintergrund über die API Open-Meteo mitgegeben werden.

In der Datenbank sieht dies dann so aus:

```
_id: ObjectId('683c4ae28e02ce23ee07d173')
see_id : "bodensee"
fischart_name : "Alet"
laenge_cm : 53
gewicht_kg : 2.8
fang_datum : 2025-06-01T12:30:00.000+00:00
fang_ort_beschreibung : "Berlingen Hafen"
koeder : "Gruyere"
bemerkungen : null
geolocation : Object
  latitude : 47.650909
  longitude : 9.131355
weather : Object
  timestamp : 2025-06-01T12:30:00.000+00:00
  temperature_celsius : 26.5
  apparent_temperature_celsius : 28
  relative_humidity_percent : 51
  precipitation_mm : 0
  weather_condition_code : 2
  wind_speed_kmh : 11.8
  wind_direction_degrees : 258
  surface_pressure_hpa : 965.7
  api_source : "Open-Meteo"
createdAt : 2025-06-01T12:43:14.145+00:00
```

## 4. Erweiterungen

Die folgenden Erweiterungen wurden über die Grundanforderungen hinaus implementiert und tragen zur Funktionalität, Effizienz und zum Wert der Anwendung bei:

### 4.1. Umfangreicher Einsatz des MongoDB Aggregation Frameworks zur Datenstrukturierung

**Beschreibung:** Die Rohdaten der fangstatistik-Collection (ein Dokument pro Fischart/See/Jahr/Typ) werden serverseitig durch das MongoDB Aggregation Framework transformiert. Die Funktion getSingleFangstatistikGroup nutzt Stages wie \$match für dynamische Filterung und insbesondere \$group zur Umstrukturierung der Daten.

Im folgenden erkläre ich die einzelnen Aggregationen im Detail.

Datei/Route: lib/server/db.js

#### 4.1.1. async function getSingleFangstatistikGroup(jahrParam, seeIdParam, typParam)

**Ziel:** Diese Funktion holt die Details für genau eine spezifische Statistikgruppe, identifiziert durch eine exakte Kombination von Jahr, See-ID und Typ, um dies dann entsprechend auf den Detailseiten anzuzeigen. Hier wird bereits eine Detailliste zurückgegeben, die den jeweiligen Parametern entspricht, so wie sie schliesslich im Frontend angezeigt wird.

```
{ $match: { jahr: jahr, see_id: seeIdParam, typ: typParam } },
{ $group: { _id: { see_id: "$see_id", jahr: "$jahr", typ: "$typ" }, fischarten: { $push:
{ fischart_id: "$fischart_id", menge_kg: "$menge_kg" } } } }
```

## 4.2. Effiziente Nutzung von distinct() für dynamische Elemente und Navigation

**Beschreibung:** Die Funktionen getUniqueLakelds, getUniqueYearsForSee und getUniqueTypesForSeeYear verwenden die MongoDB-Operation distinct(). Diese wird teilweise mit spezifischen Filterkriterien kombiniert (z.B. distinct("jahr", { see\_id: seelid })), um effizient Listen eindeutiger Werte direkt aus der Datenbank zu generieren. Diese Listen dienen der dynamischen Erstellung von Navigationslinks (z.B. für die hierarchische Statistikansicht) und der Befüllung von Dropdown-Optionen im Frontend. Dies wird vor allem verwendet, um die unstrukturierten Daten in MongoDB durch distinct nicht durcheinander zu bringen.

**Relevanz:** Zeigt den Einsatz spezialisierter Datenbankoperationen zur Optimierung der Datenbeschaffung für UI-Komponenten, anstatt alle Daten zu laden und clientseitig nach eindeutigen Werten zu filtern. Dies ist eine effiziente Methode der "Datenverwaltung".

Dateien/Routen:

- src/lib/server/db.js (Funktionen: getUniqueLakelds, getUniqueYearsForSee, getUniqueTypesForSeeYear)
- src/routes/fangstatistik/+page.server.js (nutzt getUniqueLakelds)
- src/routes/fischfangstatistik/[see\_id]/+page.server.js (nutzt getUniqueYearsForSee)
- src/routes/fischfangstatistik/[see\_id]/[jahr]/+page.server.js (nutzt getUniqueTypesForSeeYear)
- src/routes/fang-eintragen/+page.server.js (nutzt getUniqueLakelds)

## 4.3. Spezifische Datenzugriffslogik für das Single-Document Fischlexikon

**Beschreibung:** Die Datenbankfunktionen für das Fischlexikon und die Fang-Eintragen Route (getFischForCategory, getAllFischartenNamen, unterstützt durch den Helfer getFischlexikonDataDoc) sind auf die Abfrage eines einzelnen MongoDB-Dokuments zugeschnitten, in dem die gesamte Lexikoninformation mit Kategorien als Top-Level-Keys gespeichert ist. Dies beinhaltet Logik zur Extraktion von Keys als Kategoriennamen, eine Case-insensitive Suche nach Kategorien und das Sammeln aller eindeutigen Fischnamen über alle Kategorien hinweg.

**Relevanz:** Demonstriert eine angepasste Datenzugriffslogik für eine spezifische NoSQL-Datenmodellierungentscheidung, die über die Handhabung einfacher, flacher Dokumentstrukturen hinausgeht.

Dateien/Routen:

- src/lib/server/db.js (Funktionen: getFischlexikonDataDoc, getFischForCategory, getAllFischartenNamen)
- src/routes/fischlexikon/+page.server.js (nutzt getFischlexikonCategories)
- src/routes/fischlexikon/[category]/+page.js (nutzt getFischForCategory)
- src/routes/fang-eintragen/+page.server.js (nutzt getAllFischartenNamen)

## 4.4. Integration der Browser Geolocation API zur Standortermittlung

**Beschreibung:** Auf der Seite "Neuen Fang Eintragen" kann der Benutzer per Knopfdruck seinen aktuellen Standort über die Browser Geolocation API ermitteln lassen. Relevanz: Erhöht den Datenwert der Fänge durch präzise Ortsangaben, angegeben in Längen- und Breitengraden. Die Geolocation API ist ein Interface und kann durch navigator.geolocation aufgerufen werden. Es wird durch die meisten Browser unterstützt und kann verwendet werden, insofern die User die entsprechende Browser-Erlaubnis freigeschaltet haben.

Für die Entwicklung verwendete Dokumentation: [https://developer.mozilla.org/en-US/docs/Web/API/Geolocation\\_API](https://developer.mozilla.org/en-US/docs/Web/API/Geolocation_API)

Relevante Dateien/Routen:

- src/routes/fang-eintragen/+page.svelte (JavaScript-Logik für getCurrentLocation)
- src/routes/fang-eintragen/+page.server.js (Form Action empfängt Koordinaten)

#### 4.5. Anbindung der externen Open-Meteo Wetter API

**Beschreibung:** Beim Speichern eines neuen Fangs wird der durch Geolocation ermittelte Standort sowie dem Datum/der Zeit und dem Fangdatum/Uhrzeit, automatisch Wetterdaten (Temperatur, gefühlte Temperatur, relative Luftfeuchtigkeit, Niederschlag, Wettercode, Windgeschwindigkeit, Windrichtung, Oberflächenluftdruck) von der Open-Meteo API abgerufen und mit dem Fang in der Datenbank gespeichert. Leider gibt es für Seen keinen zuverlässigen Forecast von Open-Meteo. Der vorhandene Marine Forecast beschränkt sich vor allem auf Ozeane/Meere.

Für die Entwicklung verwendete Dokumentation: <https://open-meteo.com/en/docs>

**Relevanz:** Bereichert die Fangdaten um wertvolle Umweltinformationen. Die Daten sind für Nutzer im Frontend nicht sichtbar, können aber für die Fischereibewirtschaftung nützlich sein, um das Verhalten der Fische, wie zum Beispiel Jagdfenster, genauer zu erforschen.

Dateien/Routen:

- src/routes/fang-eintragen/+page.server.js (Form Action, fetch-Aufruf an Open-Meteo)

#### 4.6. Hierarische Navigation für Statistiken

**Beschreibung:** Die Fischfangstatistiken sind nicht nur als eine grosse Liste dargestellt, sondern können hierarchisch durch Auswahl von See -> Jahr -> Typ bis hin zur Detailansicht exploriert werden. Dies wird durch dynamische Routen und spezifische load-Funktionen für jede Ebene realisiert, die gezielte Datenbankabfragen auslösen. Zudem sind die aufgerufenen Daten nicht bereits so zur Verfügung gestellt, wie sie auch angezeigt werden, sondern werden via db.js dynamisch aufbereitet.

Dateien/Routen:

- src/routes/fangstatistik/+page.svelte (Seen-Links)
- src/routes/fangstatistik/[see\_id]/+page.svelte(Jahres-Links)
- src/routes/fangstatistik/[see\_id]/[jahr]/+page.svelte (Typen-Links)
- lib/server/db.js

#### 4.7. Dynamische Optionen für Dropdowns aus Datenbank

**Beschreibung:** Auf der Seite "Neuen Fang Eintragen" werden die Optionen für die Auswahl des Sees (getUniqueLakelds aus der fangstatistik-Collection) und die Vorschläge für die Fischart (getAllFischartenNamen aus der fischlexikon-Collection) dynamisch aus der Datenbank geladen. Die Daten werden also von der Datenbank gelesen und durch den User dynamisch bestimmt, anstatt diese im Frontend manuell direkt auf +page.svelte zu Schreiben.

Dateien/Routen:

- src/routes/fang-eintragen/+page.server.js (load-Funktion)
- src/routes/fang-eintragen/+page.svelte(Verwendung der geladenen Daten in <select>, <datalist>)
- src/lib/server/db.js (Funktionen getUniqueLakelds, getAllFischartenNamen)

#### 4.8. Robuste Fehlerbehandlung

**Beschreibung:** Die Anwendung implementiert Fehlerbehandlung sowohl auf der Server-Seite (in load-Funktionen und Form Actions, die error() und fail() von SvelteKit nutzen) als auch auf der Client-Seite (Anzeige von data.loadError und form.message). Formulare behalten bei Validierungsfehlern die eingegebenen Werte.

Dateien/Routen:

- Alle +page.server.js, +page.js Dateien.

- src/routes/fang-eintragen/+page.svelte (Anzeige form.message)
- Alle +page.svelte Dateien, die data.loadError anzeigen.