



**SEISMIK**

Dokumentation

Daniel Schwarz

16.11.2021



## Inhaltsverzeichnis

<b>1 Vorwort</b>	<b>4</b>
<b>2 Aufgabenstellung</b>	<b>5</b>
2.1 Anforderungen . . . . .	5
<b>3 Grobstruktur</b>	<b>6</b>
3.1 Übersicht . . . . .	6
3.2 Produktnamen . . . . .	6
3.3 Logo . . . . .	6
3.4 Funktionen . . . . .	6
<b>4 Feinstruktur</b>	<b>7</b>
4.1 Reference (Instagram) . . . . .	7
4.1.1 Users . . . . .	8
4.1.2 Posts . . . . .	9
4.1.3 Tags (und Following) . . . . .	10
4.2 Relevante Objekte . . . . .	10
4.2.1 Users . . . . .	10
4.2.2 Photos . . . . .	12
4.2.3 Tags . . . . .	14
4.3 Beziehungen . . . . .	15
4.3.1 posten . . . . .	15
4.3.2 liken . . . . .	16
4.3.3 comments . . . . .	16
4.3.4 having . . . . .	16
4.3.5 follows . . . . .	17
4.4 Typische Operationen . . . . .	17
4.5 weitere relevante Informationen . . . . .	18
4.5.1 Datenmengen . . . . .	18
4.5.2 Gesetzliche Bestimmungen . . . . .	18
4.5.3 Sicherung der Datenbank . . . . .	18
4.5.4 Zeichensatz . . . . .	18
<b>5 Entity Relationship-Modell</b>	<b>19</b>
5.1 Gesamtdarstellung . . . . .	19
5.2 Detaildarstellung . . . . .	19
5.2.1 USERS . . . . .	19
5.2.2 PHOTOS . . . . .	20



5.2.3 TAGS . . . . .	20
<b>6 Relationenmodell</b>	<b>21</b>
<b>7 Normalisierung</b>	<b>22</b>
<b>8 Datenbankobjekte</b>	<b>23</b>
8.1 Tabellen . . . . .	23
8.2 Dummy-Datensätze . . . . .	23
<b>9 Datenmanipulation</b>	<b>24</b>
9.1 Foto posten . . . . .	24
9.2 Foto löschen . . . . .	24
9.3 Liken . . . . .	24
9.4 Unliken . . . . .	24
9.5 Folgen . . . . .	25
9.6 Unfollow . . . . .	25
<b>10 Datenabfrage</b>	<b>26</b>
10.1 Bilder des Users anzeigen . . . . .	26
10.2 Kommentare eines Bildes . . . . .	26
10.3 Top 5 liked Fotos . . . . .	26
<b>11 Menügestaltung</b>	<b>27</b>
11.1 Ordner- und Datei-Struktur . . . . .	27
11.2 Menüübersicht . . . . .	28
11.2.1 Hauptmenü . . . . .	29
11.2.2 Untermenü . . . . .	30



## 1 Vorwort

Diese Dokumentation wurde für mein Projekt bei Cimdata mit OracleSQL erstellt.

Aufgabe ist es, eine Datenbank anzulegen und mit Hilfe von SQL-Scripten via dem cl-tool sqlplus/sqlcl ein kleines “Programm” zu erstellen. Zusätzlich ist eine Dokumentation sowie Präsentation gewünscht.

Die SQL-Scripte sollen:

- eine Datenbank anlegen
- die erforderlichen Tabellen erstellen
- Dummy Daten in die Tabellen einfügen
- eine Menüstruktur zur wiederholten Ausführung anbieten
- man soll folgendes im Menü auswählen können
  - Datenbank aufsetzen
  - Datensätze löschen
  - Datensätze anlegen
  - Datenabfragen realisieren
  - Datenbank löschen

Ich habe hierbei mit Gnu/Linux - Fedora Silverblue<sup>1</sup> - gearbeitet. Um den Oracle SQL Server aufzusetzen benutze ich Podman - ein Containertool, verwendet um den Server in einem Container aufzusetzen.

Zur Dokumentation habe ich Markdown benutzt und dieses mit dem Tool Eisvogel in eine PDF-Datei gerendert.

**Listing 1:** Eisvogel cl-Befehl

```
1 pandoc documentation.md -o documentation.pdf --from markdown --template eisvogel -V lang=de --shift-heading-level-by=-1 --toc --number-sections --listings
```

---

<sup>1</sup> Immutable OS

## 2 Aufgabenstellung

Entwickeln Sie eine Datenbank zu einem von Ihnen gewählten Thema mit den Schwerpunkten: Konzeptueller und physischer Entwurf der Datenbank.

### 2.1 Anforderungen

Innerhalb der Projektwoche (Montag bis Donnerstag) soll eine Aufgabe oder abgegrenzte Teilaufgabe bearbeitet werden. Hierfür müssen folgende Projekt-Schwerpunkte umgesetzt werden:

- Konzeptueller Entwurf der Datenbank, deren Ergebnis in der Dokumentation festgehalten wird,
- Physischer Entwurf der Datenbank, deren Ergebnis durch Skripte realisiert wird und
- Präsentation des Projektes

Die Bearbeitung des Projektes kann als Einzel- oder Gruppenarbeit, bestehend aus 2 Teilnehmern erfolgen. Der Dozent ist prinzipiell der Gruppenleiter, dessen Eingreifen in die Geschicke der Gruppe nur in Ausnahmefällen passiert, um beispielsweise das Projekt voranzubringen.

Mit Beginn des Projektes muss sich jeder Studierende entschieden haben, ob das Projekt als Einzel- oder Gruppenarbeit durchgeführt wird. Ein nachträgliches Ändern des Teams ist während der Projektphase nicht möglich.



## 3 Grobstruktur

### 3.1 Übersicht

Der Kunde wünscht, eine europäische Konkurrenz-Entität für Instagram zu etablieren. Hierfür soll verstärkt auf Anonymität, No-Tracking und Datenschutz für zukünftige Benutzer gesetzt werden. Zudem soll es Werbefrei gestaltet werden. User können Geld verdienen in dem Sie bestimmte Kriterien erfüllen.

Das Produkt soll in einer ersten Version als Prototyp dienen, um notwendige Schnittstellen, zu bereits vorhanden Backendsystemen des Kunden auszubauen und anpassen zu können. Später wird das Projekt dann im Umfang ausgebaut und auf verschiedenen Frontendapps verteilt. Angedacht sind Mobile-apps, eine Web-app als SaaS sowie eine Desktop-app für die gängigen Betriebssysteme.

### 3.2 Produktnamen

Für das Projekt habe ich den Namen “**SEISMIK**” gewählt. In der Überzeugung das wir mit diesem Produkt den Markt der Social-Media-Apps “erschüttern” werden.

### 3.3 Logo

Das Logo und die Corporate Identity habe ich in Abstimmung mit dem Auftraggeber von einem Grafik-Designer entwickeln lassen und lizenziert.

### 3.4 Funktionen

Benutzer sollen sich Registrieren können und anschließend ein- bzw. ausloggen. Sie können dann Posts erstellen in deren Inhalt es sich - in der ersten Ausbaustufe - vor allem um Bilder handelt. Die Benutzer sollen Posts von anderen Benutzern liken und kommentieren können. Benutzer können andere Benutzern folgen.

## 4 Feinstruktur

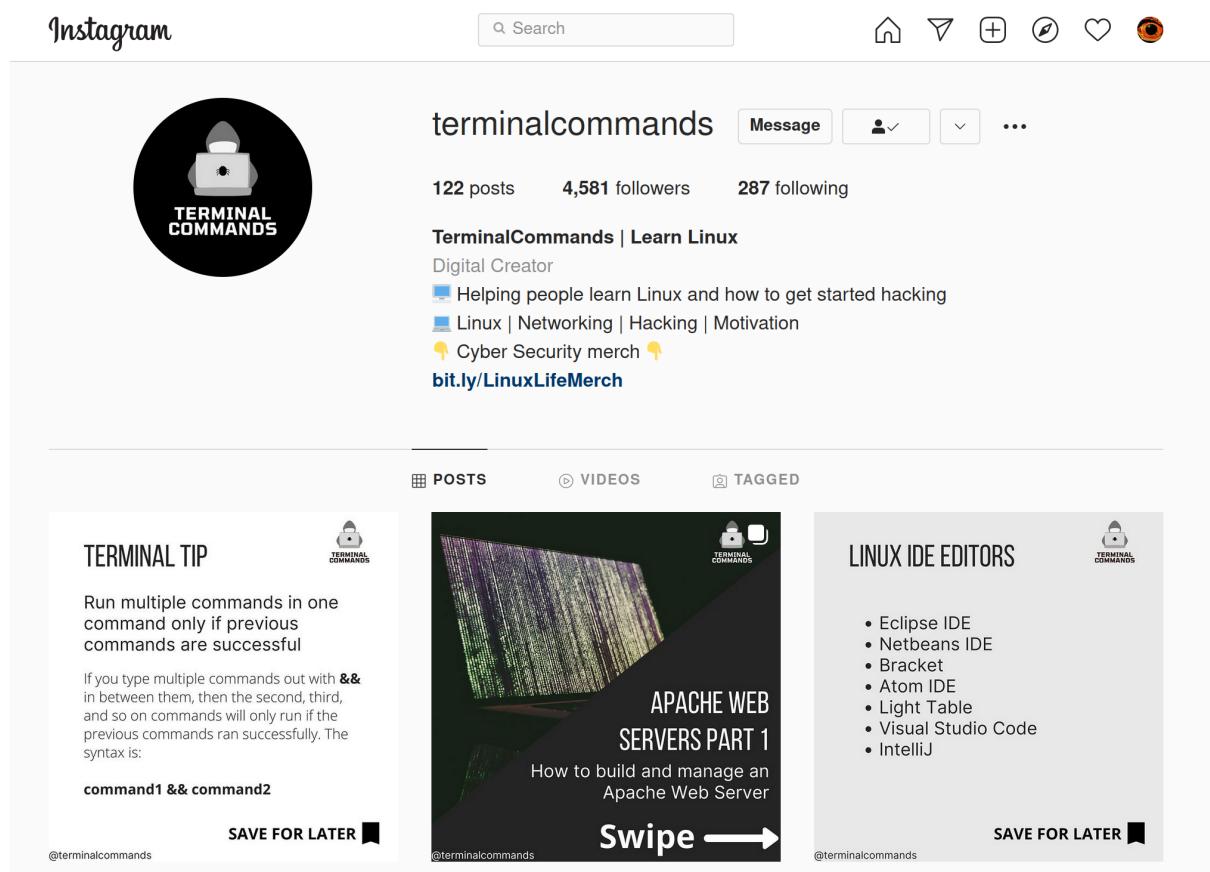
### 4.1 Reference (Instagram)

Instagram ist ein soziales Netzwerk, ähnlich Twitter oder Facebook. Der Fokus liegt hier vor allem auf Video- und Foto-Sharing.

Mit mehr als 20 Mio. Nutzern allein in Deutschland und 150 Mio. Weltweit zählt Instagram zu einem der größeren Vertreter der Social-Media-Apps.

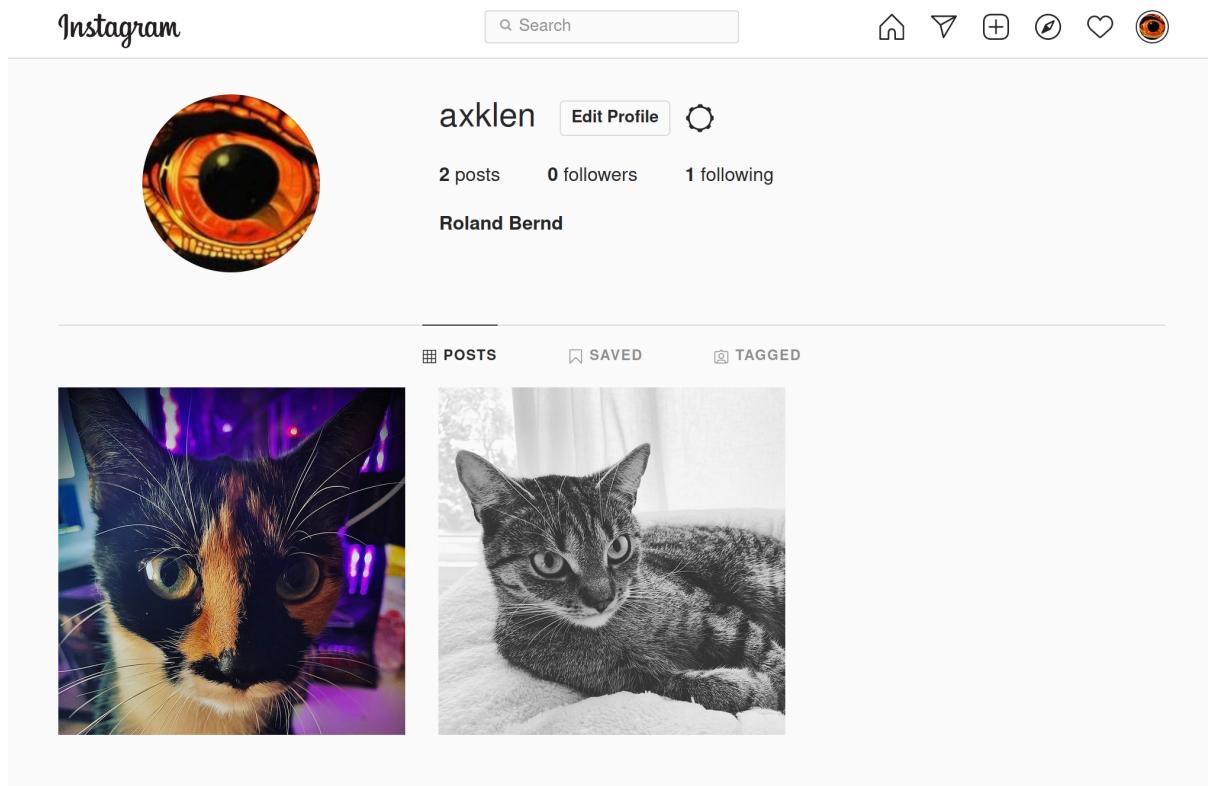
---

Man kann es sich wie eine Art Online-Fotoalbum vorstellen, in dem man Einblicke in sein Leben geben kann. Es ist auf allen gängigen Plattformen erhältlich und wird auch gern von Unternehmen zur Reichweitensteigerung genutzt.



**Abbildung 1:** Instagram - Übersicht

#### 4.1.1 Users



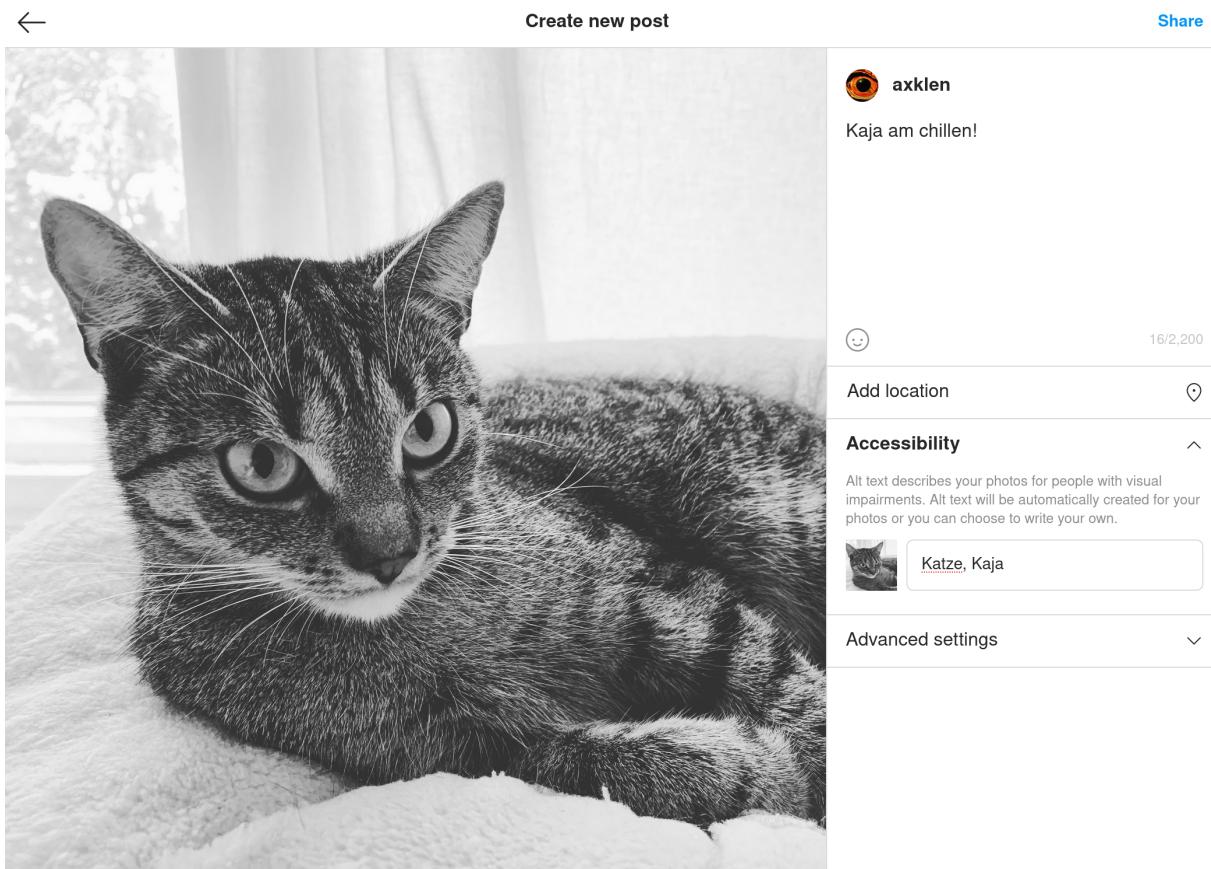
**Abbildung 2:** Instagram - Useraccount

Benutzer laden Bilder hoch, die sie durch Tags (Hashtags) kategorisieren können. Diese Bilder können in Ihrer Größe verändert werden und man kann verschiedene Farbfilter auf sie Anwenden.

#### Funktionsübersicht

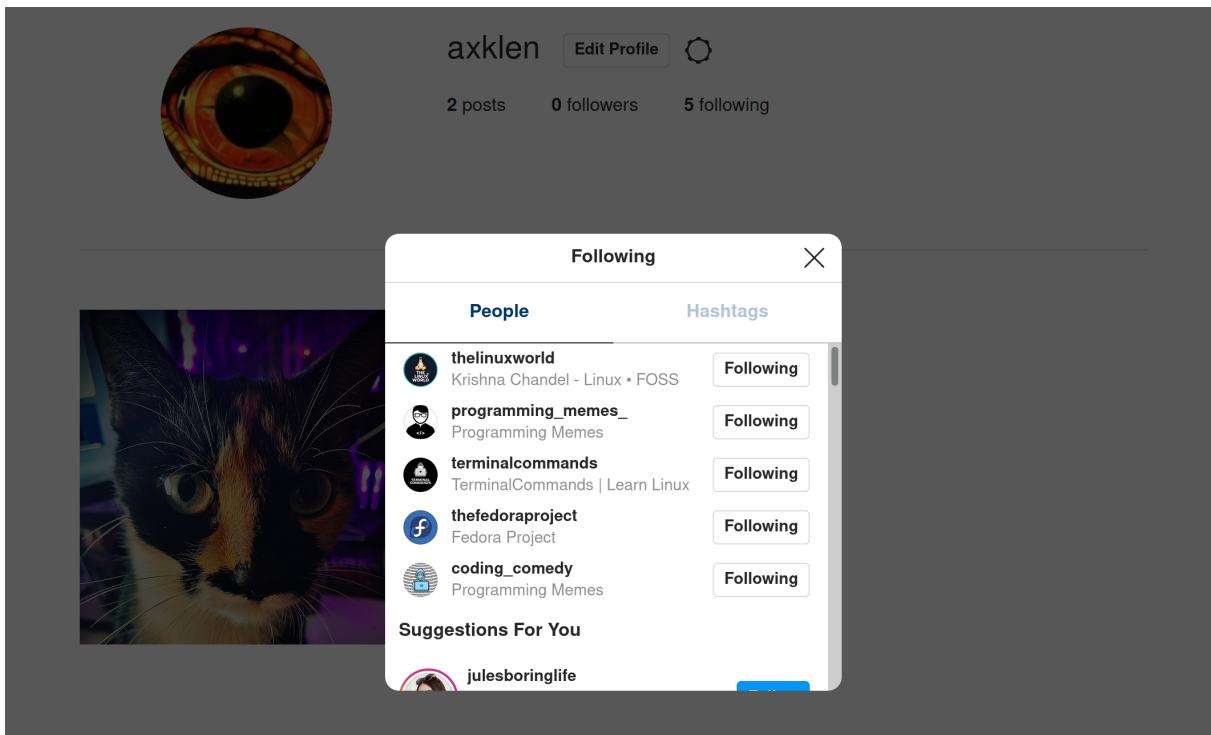
- Hochladen von Bildern aus dem Album oder direkte Verknüpfung mit der Kamera des Smartphones oder Tablets
- Hochladen oder Drehen von Videos
- Geotag-Funktion, über die eine eigene Fotokarte eines Nutzers angelegt werden kann
- Austausch über Fotos mit anderen Usern
- Like-Funktion in Form eines Herzen
- Kommentar-Funktion, mit der Möglichkeit Hashtags zu verwenden

#### 4.1.2 Posts



**Abbildung 3:** Instagram - Post eines Fotos

#### 4.1.3 Tags (und Following)



**Abbildung 4:** Instagram - Following und Hashtags

## 4.2 Relevante Objekte

Aus der Analyse von Instagram ergeben sich die nachfolgend beschriebenen relevanten Objekte für unser Projekt SEISMIK.

### 4.2.1 Users

Das sind die Benutzeraccounts zur Identifizierung eines Benutzers.

- Benutzername ist unique und nicht leer (min. 4 Zeichen)
- Passwort ist nicht leer (min. 8 Zeichen)
- beim erstellen des Benutzers wird ein Timestamp erstellt

<b>id</b>	Eindeutige ID zur internen Identifizierung der User
Wertebereich	Ganze Zahlen
Werte-Einschränkung	Nein



<b>id</b>	Eindeutige ID zur internen Identifizierung der User
Wertevergabe	System / Auto_Increment
Default-Wert	Nein
NULL-Wert-erlaubt	Nein
Duplikate erlaubt	Nein
Primärschlüssel	Ja
Sonst	automatisch vom System verwaltet über Constraint
Beispielwerte	1, 2, 3, ...

<b>username</b>	Eindeutiger vom Benutzer gewählter Identifier
Wertebereich	Text
Werte-Einschränkung	min. 4 Zeichen
Wertevergabe	Benutzereingabe
Default-Wert	Nein
NULL-Wert-erlaubt	Nein
Duplikate erlaubt	Nein
Primärschlüssel	Nein
Sonst	-
Beispielwerte	'Roland', '11KidShredder', 'user(dummy)'

<b>passwd</b>	vom Benutzer gewähltes Passwort gehashed in SHA2
Wertebereich	Text
Werte-Einschränkung	min. 8 Zeichen
Wertevergabe	Benutzereingabe
Default-Wert	Nein
NULL-Wert-erlaubt	Nein
Duplikate erlaubt	Ja
Primärschlüssel	Nein



---

<b>passwd</b>	vom Benutzer gewähltes Passwort gehashed in SHA2
Sonst	-
Beispielwerte	'12345678', '12fdf2e123', 'sdf-?. .ddds'

---

---

<b>created_at</b>	vom System automatisch erstellter Timestamp
Wertebereich	Datum & Zeit
Werte-Einschränkung	Nein
Wertevergabe	System, bei Erstellung des Accounts
Default-Wert	Ja, sysdate
NULL-Wert-erlaubt	Ja
Duplikate erlaubt	Ja
Primärschlüssel	Nein
Sonst	-
Beispielwerte	'11.11.2021 23:59:59' ...

---

#### 4.2.2 Photos

Dies sind die Photos die von Usern gepostet werden

- Das Foto wird vorerst als url gespeichert
- beim erstellen des Fotos wird ein Timestamp erstellt

---

<b>id</b>	Eindeutige ID zur internen Identifizierung der Bilder
Wertebereich	Ganze Zahlen
Werte-Einschränkung	Nein
Wertevergabe	System / Auto_Increment
Default-Wert	Nein
NULL-Wert-erlaubt	Nein
Duplikate erlaubt	Nein
Primärschlüssel	Ja



---

<b>id</b>	Eindeutige ID zur internen Identifizierung der Bilder
Sonst	automatisch vom System verwaltet über Constraint
Beispielwerte	1, 2, 3, ...

---

---

<b>image_url</b>	Verlinkt zum geposteten Bild eines Benutzers
Wertebereich	Text
Werte-Einschränkung	Nein
Wertevergabe	Benutzereingabe
Default-Wert	Nein
NULL-Wert-erlaubt	Nein
Duplikate erlaubt	Ja
Primärschlüssel	Nein
Sonst	-
Beispielwerte	'img/katze.jpg' ...

---

---

<b>created_at</b>	vom System automatisch erstellter Timestamp
Wertebereich	Datum & Zeit
Werte-Einschränkung	Nein
Wertevergabe	System, bei Erstellung des Accounts
Default-Wert	Ja, sysdate
NULL-Wert-erlaubt	Ja
Duplikate erlaubt	Ja
Primärschlüssel	Nein
Sonst	-
Beispielwerte	'11.11.2021 23:59:59' ...

---



### 4.2.3 Tags

Jedes Bild kann verschiedenen Tags zugeordnet werden.

- Tags sind Einzigartig
- beim erstellen des Tags wird ein Timestamp erstellt

<b>id</b>	Eindeutige ID zur internen Identifizierung der Bilder
Wertebereich	Ganze Zahlen
Werte-Einschränkung	Nein
Wertevergabe	System / Auto_Increment
Default-Wert	Nein
NULL-Wert-erlaubt	Nein
Duplikate erlaubt	Nein
Primärschlüssel	Ja
Sonst	automatisch vom System verwaltet über Constraint
Beispielwerte	1, 2, 3, ...

<b>tag_name</b>	Bezeichner für die verschiedenen Tags der Bilder
Wertebereich	Text
Werte-Einschränkung	Nein
Wertevergabe	Benutzereingabe
Default-Wert	Nein
NULL-Wert-erlaubt	Nein
Duplikate erlaubt	Nein
Primärschlüssel	Nein
Sonst	-
Beispielwerte	'cat', 'drachenlordImKnast', 'WM2020' ...



<b>created_at</b>	vom System automatisch erstellter Timestamp
Wertebereich	Datum & Zeit
Werte-Einschränkung	Nein
Wertevergabe	System, bei Erstellung des Accounts
Default-Wert	Ja, sysdate
NULL-Wert-erlaubt	Ja
Duplikate erlaubt	Ja
Primärschlüssel	Nein
Sonst	-
Beispielwerte	'11.11.2021 23:59:59' ...

---

### 4.3 Beziehungen

Aus der Analyse von Instagram ergeben sich die nachfolgend beschrieben Beziehungen für unser Projekt SEISMIK.

#### 4.3.1 posten

Ist eine Binäre Beziehung zwischen Users und Photos.

- Jeder User kann ein oder mehrere Fotos posten
- Jedes Foto wird von genau einem User gepostet

<b>Beteiligung</b>		<b>Kardinalität</b>
Users	0	N
	Jeder User muss kein Foto posten	Jeder User kann mehrere Fotos posten
Photos	1	1
	Jedes Foto muss von einem User gepostet werden	Jedes Foto wird von höchstens einem User gepostet

---

#### 4.3.2 liken

Ist eine Binäre Beziehung zwischen Users und Photos.

- Jeder User kann ein oder mehrere Fotos liken
  - Jedes Photo kann von mehreren Usern geliked werden
- 

<b>Beteiligung</b>		<b>Kardinalität</b>
Users	0	N
	Jeder User muss kein Foto liken	Jeder User kann mehrere Fotos liken
Photos	0	M
	Jedes Foto muss nicht von einem User geliked werden	Jedes Foto kann von mehreren Usern geliked werden

#### 4.3.3 comments

Ist eine Binäre Beziehung zwischen Users und Photos.

- Jeder User kann ein oder mehrere Fotos kommentieren
  - Jedes Photo kann von mehreren Usern kommentiert werden
- 

<b>Beteiligung</b>		<b>Kardinalität</b>
Users	0	N
	Jeder User muss kein Foto kommentieren	Jeder User kann mehrere Fotos kommentieren
Photos	0	M
	Jedes Foto muss nicht von einem User kommentiert werden	Jedes Foto kann von mehreren Usern kommentiert werden

#### 4.3.4 having

Ist eine Binäre Beziehung zwischen Tags und Photos.

- Jeder Tag muss einem oder mehreren Fotos zugeordnet sein
- Jedes Photo kann mit mehreren Tags gekennzeichnet werden

	<b>Beteiligung</b>	<b>Kardinalität</b>
Tags	1  Jeder Tag muss einem Foto zugeordnet sein	N  Jeder Tag kann mehreren Fotos zugeordnet sein
Photos	0  Jedes Foto muss nicht mit einem Tag gekennzeichnet werden	M  Jedes Foto kann mit mehreren Tags gekennzeichnet werden

#### 4.3.5 follows

Ist eine Unäre Beziehung von Users

- Jeder Follower kann mehreren Followees folgen
- Jeder Followee kann mehrere Follower besitzen

	<b>Beteiligung</b>	<b>Kardinalität</b>
Users	0  Jeder follower muss keinem followee folgen	N  Jeder follower kann mehreren followees folgen
Users	0  Jeder followee muss keinen follower besitzen	M  Jeder followee kann von mehreren followern gefolgt werden

### 4.4 Typische Operationen

- Benutzer login
- Anzeige Kommentare eines Bildes
- Ausgabe aller Followees des Benutzers
- Ausgabe der Followers des Benutzers
- Ausgabe aller likes eines Bildes

## 4.5 weitere relevante Informationen

Im nachfolgend sind weitere relevanten Informationen aufgelistet, die bei der Erstellung der Datenbank und Auswahl des Datenbanksystems zu beachten sind.

### 4.5.1 Datenmengen

Über das Datenvolumen der Datenbank lässt sich an dieser Stelle noch keine ganz genaue Aussage treffen. Unter anderem fehlen Informationen, in welchem Format und Auflösung die Bilder später gespeichert werden und ob auch noch Videos und andere Metadaten hinzu kommen werden.

Es ist allerdings davon auszugehen das die Datenbank schnell mehrere TB groß werden kann.

### 4.5.2 Gesetzliche Bestimmungen

Da wir uns im Internet bewegen sind die jeweiligen Rechtsbestimmungen aller Länder zu beachten. Besonders in Hinsicht auf Datenschutz und möglicher (Anti-) Zensurgesetze.

Wir werden uns in der ersten Phasen nur auf den europäischen Markt konzentrieren und nehmen die Europäische Datenschutz-Grundverordnung sowie nationale Regelungen als mindestmaß für unser Konzept.

Zudem sind strafrechtliche Bestimmungen der einzelnen Länder zu beachten.

### 4.5.3 Sicherung der Datenbank

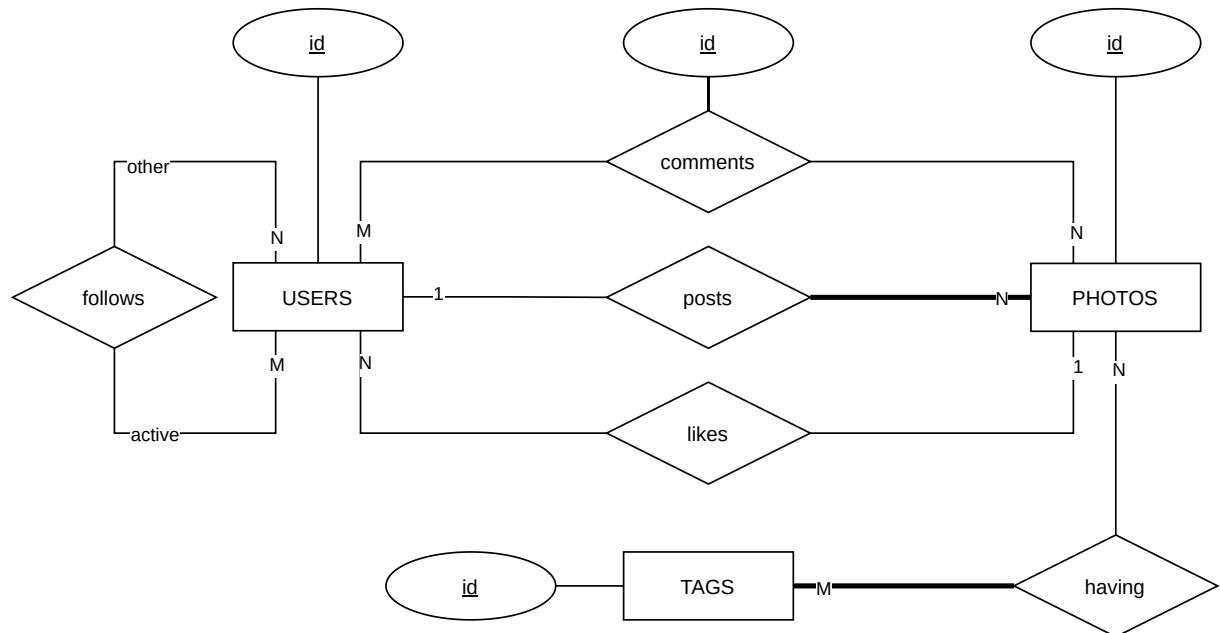
Wir können uns keine Downtimes der Datenbank erlauben, da wir uns auf einem globalen Markt bewegen. Die Datenbank wird als micro-service in einem orchestrierten Container-cluster laufen. Sie wird durch Replizierung gegen einen Ausfall gesichert und jeweils asynchron auf externe Sicherungshubs verteilt.

### 4.5.4 Zeichensatz

Als Zeichensatz werden wir UTF-8 einsetzen.

## 5 Entity Relationship-Modell

### 5.1 Gesamtdarstellung

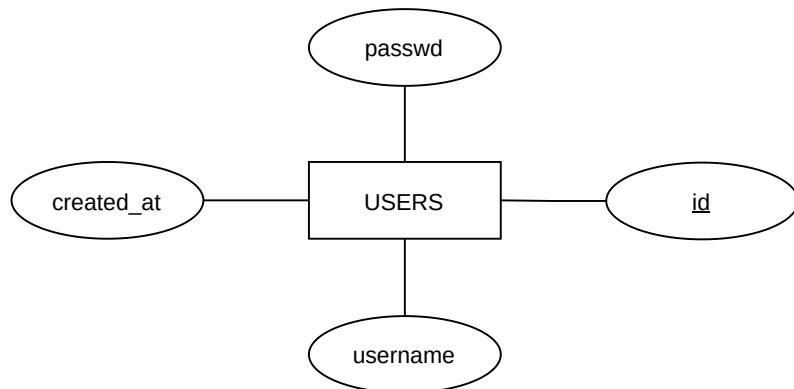


**Abbildung 5:** Gesamtübersicht SEISMIC

### 5.2 Detaildarstellung

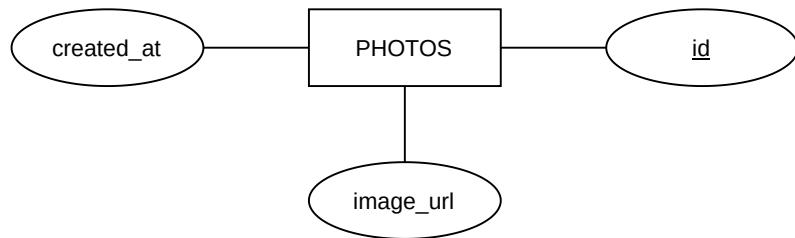
Jede Entität wird nachfolgend detailliert mit den dazugehörigen Attributen dargestellt.

#### 5.2.1 USERS



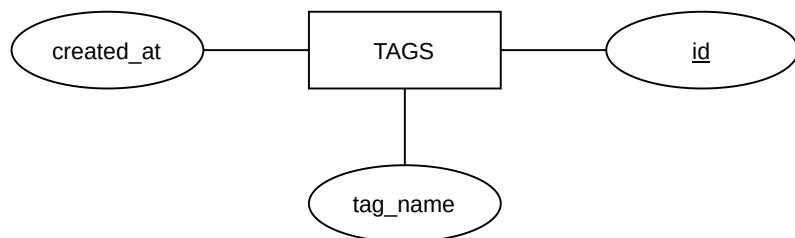
**Abbildung 6:** SEISMIC - Entität USERS

### 5.2.2 PHOTOS



**Abbildung 7:** SEISMIK - Entität PHOTOS

### 5.2.3 TAGS



**Abbildung 8:** SEISMIK - Entität TAGS



## 6 Relationenmodell

Users ( `id` , `passwd` `NN` , `username` `NN` , `created_at` )

Photos ( [id](#) , [image\\_url](#) NN , [created\\_at](#) , [user\\_id](#) NN )

FK: Photos.user\_id => PK: Users.id

Tags ( [id](#) , [tag\\_name](#) NN , [created\\_at](#) )

Comments ( [id](#) , comment\_text NN , user\_id NN , photo\_id NN , created\_at )

FK: Comments.user\_id => PK: Users.id

FK: Comments.photo\_id => PK: Photos.id

Likes ( user\_id , photo\_id , created\_at )

FK: Likes.user\_id => PK: Users.id

FK: Likes.photo\_id => PK: Photos.id

Photo\_tags (photo\_id, tag\_id)

FK: Photo\_tags.photo\_id => PK: Photos.id

FK: Photo\_tags.tag\_id => PK: Tags.id

Follows ( follower\_id , followee\_id )

FK: Follows.follower\_id => PK: Users.id

FK: Follows.followee\_id => PK: Users.id

**Abbildung 9:** SEISMIC - Relationenmodell

## 7 Normalisierung

An dieser Stelle sollen nur die ersten 3 Normalformen betrachtet werden.

1. Normalform Alle Attribute weisen nur atomare Werte auf.
2. Normalform: Da die 1. Normalform erfüllt ist und alle Primärschlüssel aus künstlichen Schlüsseln gebildet werden, liegt die 2. Normalform vor.
3. Normalform: Die 2. Normalform liegt vor und es existieren keine Nichtschlüsselattribute, die transitiv vom Primärschlüssel abhängen. Somit ist die 3. Normalform ebenfalls gegeben.



## 8 Datenbankobjekte

Zum Erstellen der Datenbankobjekte wird ein Benutzer benötigt, der über ausreichende Rechte verfügt.

### 8.1 Tabellen

- Erstellung aller vorher aufgelisteten Tabellen von SEISMIK
- löschen bereits vorhandener Tabellen

Script: setup/db\_table\_install.sql

### 8.2 Dummy-Datensätze

- leeren bereits vorhandener Tabellen
- einfügen der Dummy-Datensätze in jede Tabelle
- speichern (COMMIT) der Transaktion

Skript: setup/insert\_data.sql

## 9 Datenmanipulation

Die Manipulation der Daten erfolgt durch interaktive Eingabe der Werte.

### 9.1 Foto posten

Es wird ein neues Foto (derzeit noch link) des angemeldeten Benutzers(Login) in der Tabelle `photos` gespeichert. Der Benutzer wird zum Datenupload (derzeit noch url) des Fotos aufgefordert. Der Primärschlüssel wird vom DBMS selbst verwaltet. Am Ende werden die Änderungen gespeichert (Committed)

Skript: update/post.sql

### 9.2 Foto löschen

Es wird ein ausgewähltes Foto (derzeit noch link) des angemeldeten Benutzers(Login) aus der Tabelle `photos` gelöscht. Der Benutzer wählt die ID des Fotos das er löschen möchte aus. Dann wird dass Foto in der Tabelle `photos` gelöscht. Anschließend werden in allen Beziehungen (`likes`, `comments`, `photo_tags`) die jeweiligen zugehörigen Datensätze gelöscht. Am Ende werden die Änderungen gespeichert (Committed).

Skript: update/post.sql

### 9.3 Liken

Es werden dem Benutzer alle Fotos die er noch nicht geliked hat angezeigt. Der eingeloggte Benutzer wählt eine Foto ID aus (`photos`). Dann wird in der Tabelle `likes` die ID des eingeloggten Benutzers sowie die ID des gewählten Fotos eingetragen. Am Ende werden die Änderungen gespeichert (Committed). Die beiden gesetzten ID sind jeweils Fremdschlüssel mit Referenz zu `users` und `photos`.

Skript: update/like.sql

### 9.4 Unlikken

Dem Benutzer werden alle Fotos angezeigt die er geliked hat. Er wird aufgefordert eine ID eines Fotos das er unliken will zu wählen. Es werden dann in der Tabelle `likes` die beiden Fremdschlüssel `user_id` und `photo_id` gelöscht. Diese Transaktion wird anschließend beendet (Committed). Beide Werte sind jeweils Fremdschlüssel mit Referenz auf `users` und `photos`.

Script: update/unlike.sql



## 9.5 Folgen

Am Anfang erhält der Benutzer eine Übersicht der Nutzer denen er Folgen kann (der Benutzer und bereits gefolgte Benutzer werden nicht angezeigt). Er wählt dann die `id` des Benutzers dem er folgen will. Es werden dann in der Tabelle `follows` jeweils die ID des Benutzers `follower_id` sowie die ID des ausgewählten Benutzers `followee_id` eingetragen. Die Transaktion wird anschließend beendet (Committed). Jede ID ist jeweils ein Fremdschlüssel.

Script: update/follow.sql

## 9.6 Unfollow

Der Benutzer wählt aus einer Liste der bereits gefolgten Benutzern die `followee_id` aus. Anschließend wird in der Tabelle `follows` der Datensatz mit der ausgewählten `followee_id` und der ID des Benutzers `follower_id` gelöscht. Am Ende werden die Änderungen gespeichert (Committed). Die beiden gesetzten ID sind jeweils Fremdschlüssel mit einer Referenz zu `users`.

Script: update/unfollow.sql



## 10 Datenabfrage

Eine Auflistung der derzeitigen Datenabfragen sollen nachfolgend vorgestellt werden.

### 10.1 Bilder des Users anzeigen

Welche Bilder hat der User bereits gepostet mit Gesamtzahl der Likes und Comments je Bild.

Script: read/userphotos.sql

### 10.2 Kommentare eines Bildes

Der User erhält eine Auswahl aller Bilder mit Kommentaren > 0. Er wählt dann eines via ID aus. Es wird dem User dann eine Tabelle mit allen Kommentaren zu diesem Bild präsentiert, absteigend sortiert nach gepostetem Datum und wer diesen post erstellt hat.

Script: read/neusteK.sql

### 10.3 Top 5 liked Fotos

Eine Übersicht der Top 5 gelikten Fotos (Fotos mit den 5 höchsten like werten[doppelte Platzierungen nicht ausgeschlossen]).

Script: read/top5.sql

## 11 Menügestaltung

Die Verwaltung der Datenbankobjekte, die Datenmanipulation und die Datenabfrage kann menügesteuert ausgeführt werden. Die Menügestaltung wird mit den Möglichkeiten, die `sqlcl` bietet, realisiert.

### 11.1 Ordner- und Datei-Struktur

Die Menügestaltung basiert auf der nachfolgend dargestellten Struktur.

```
└─ toolbox └─ ~/Doc/IT3/projects_IT33/seismik/scripts on ↵ main [?] [O github.com/Axklen/projects_IT33]
  └─ ls --tree

    └── read/
        ├── menu.sql
        ├── neusteK.sql
        ├── top5.sql
        └── userphotos.sql

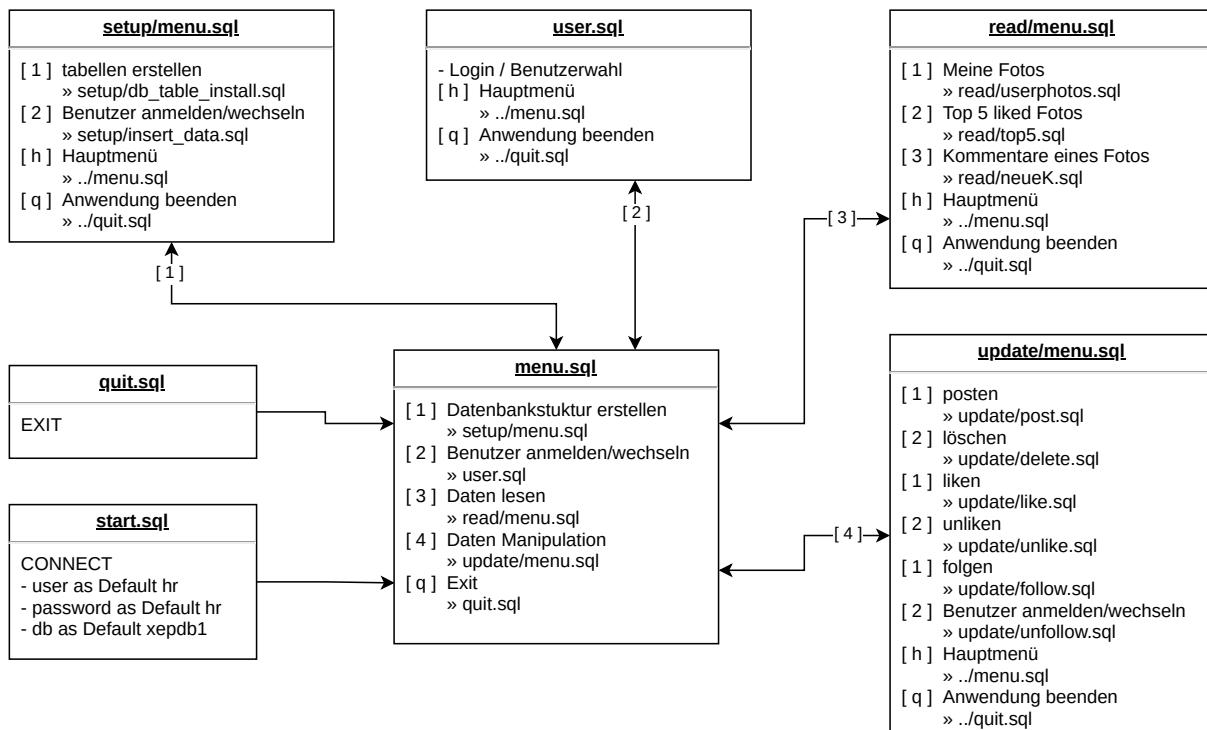
    └── setup/
        ├── db_table_install.sql
        ├── insert_data.sql
        └── menu.sql

    └── update/
        ├── delete.sql
        ├── follow.sql
        ├── like.sql
        ├── menu.sql
        ├── post.sql
        ├── unfollow.sql
        └── unlike.sql

    └── asciiArt
    └── menu.sql
    └── quit.sql
    └── start.sql
    └── user.sql
```

**Abbildung 10:** Ordner-/Dateistruktur

## 11.2 Menüübersicht



**Abbildung 11:** Menügestaltung

Zum Starten wird das shellscript: `start.sh` ausgeführt. Es sorgt dafür, dass das Programm `SQLcl` ohne eine Anmeldung an die Datenbank gestartet wird. Nach erfolgreichem Start der Anwendung wird das Skript: `start.sql` ausgeführt.

Im Skript: `start.sql` werden die folgenden Substitutionsvariablen generiert:

- `usr` => Name des Datenbankbenutzers (Default: `hr`)
- `passwd` => Passwort des Datenbankbenutzers (Default: `hr`)
- `db` => Datenbankname der Datenbank (Default: `XEPDB1`)

Statt dem vorgeschlagenen Default-Wert kann der Benutzer interaktiv einen neuen Wert vergeben. Der Benutzer wird danach mit der Datenbank verbunden und anschließend das Hauptmenü durch Aufrufen des Skriptes: `menu.sql` präsentiert.

**11.2.1 Hauptmenü****Abbildung 12:** Hauptmenü

Im Hauptmenü kann zwischen folgenden Menüpunkten gewählt werden, deren Auswahl in den jeweiligen eckigen Klammerpaar angegeben wird:



Menüpunkt	Bemerkung	Skript
Datenbankstruktur bearbeiten	Ruft das Untermenü: Datenbankstruktur auf	<a href="#">setup/menu.sql</a>
Benutzerauswahl / -wechsel	Ruft das Script Login auf	<a href="#">user.sql</a>
Lesen spezifischer Datensätze ändern von Datensätzen	Ruft das Untermenü: Daten lesen auf Ruft das Untermenü: Daten ändern auf	<a href="#">read/menu.sql</a> <a href="#">update/menu.sql</a>

---

### 11.2.2 Untermenü