

# Datenmodell für Webanwendung

```
---Add commentMore
actions title: Data
Model
parent: Technical
Docs nav_order: 3
---
```

```
[Kaan Baskaya]
{: .label }
```

```
# Data model
{: .no_toc }
```

Our web application focuses on enabling friendships between students by using personality based matching. The core functionalities are: user registration, login, storing personality test results, swiping other users, and matches when both users like each other.

The following database tables support these functions: user, university, personality\_result, swipe, and match.

Below is the breakdown of the relevant attributes, their types, and constraints.

```
# Table for Users
```

```
<details open markdown="block">
<summary>- User/</summary>
<pre>
```

```

"user_id"    INTEGER PRIMARY KEY,
"uni_id"
            INTEG
ER, "name"  TEXT
NOT NULL,
"email"      TEXT UNIQUE NOT NULL,
"password"   TEXT
NOT NULL, "age"
            INTEGER
,
"created_at"  DATE DEFAULT CURRENT_DATE,
FOREIGN KEY ("uni_id") REFERENCES
university("uni_id")
</pre>
</details>

```

1/6

## # Table for Universities

```

<details open markdown="block">
<summary>- University</summary>
<pre>
"uni_id"    INTEGER
PRIMARY KEY, "name"
            TEXT NOT
NULL,
"short_nam
e"          TEXT, "location" TEXT
</pre>
</details>

```

## # Table for Personality Test Results

```

<details open markdown="block">

```

<summary>- Personality\_Result</summary>

<pre>

```
"result_id"    INTEGER
PRIMARY KEY, "user_id"
              INTEGER,
"vec_ei"       FLOAT,
"vec_sn"       FLOAT,
"vec_tf"       FLOAT,
              "vec_jp"    FLOAT, "mbti_type"    TEXT, "completed_at" DATETIME
FOREIGN KEY ("user_id") REFERENCES
user("user_id")
```

</pre>

</details>

# Table for Swipes

<details open markdown="block">

<summary>- Swipe</summary>

<pre>

```
"swipe_id"      INTEGER PRIMARY KEY,
"swiper_id"     INTEGER, "swiped_id"      INTEGER,
"swipe_type"    TEXT, -- "like"
or "dislike" "swiped_at"
                DATE,
"match_generated" BOOLEAN DEFAULT
FALSE, FOREIGN KEY ("swiper_id")
REFERENCES user("user_id"),
```

```
FOREIGN KEY ("swiped_id") REFERENCES
user("user_id")
```

</pre>

</details>

# Table for Matches

```
<details open markdown="block">
<summary>- Match/</summary>
<pre>Add commentMore
  actions "match_id"
          INTEGER
        PRIMARY KEY,
        "user1_id"
        INTEGER,
        "user2_id"
        INTEGER,
        "compatibility
        " FLOAT,
        "status" TEXT, -- e.g., "active",
        "expired" "matched_at" DATE,
        FOREIGN KEY ("user1_id") REFERENCES
        user("user_id"), FOREIGN KEY ("user2_id")
        REFERENCES user("user_id")
</pre>
</details>
```

**\*\*Constraints:\*\***

Most fields use NOT NULL where appropriate to ensure data integrity. Foreign keys are used to enforce relationships between users and universities, test results, swipes, and matches.

kannst du diese github datei auf deutsch machen wir wollen unser projekt doch auf deutsch erstellen

Natürlich! Hier ist deine GitHub-Markdown-Datei vollständig ins Deutsche übersetzt, inklusive technischer Begriffe und Beschreibungen:

3/6

*[Kaan Baskaya]*

*{: .label }*

*# Datenmodell*

*{: .no\_toc }*

Unsere Webanwendung zielt darauf ab, Freundschaften zwischen Studierenden durch persönlichkeitsbasierte Vorschläge zu ermöglichen. Die Kernfunktionen umfassen: Nutzerregistrierung, Login, Speicherung der Ergebnisse des Persönlichkeitstests, Wischen anderer Nutzer sowie Matches, wenn sich zwei Personen gegenseitig liken.

Die folgenden Datenbanktabellen unterstützen diese Funktionen: nutzer, universität, persönlichkeitsergebnis, swipe und match.

Nachfolgend sind die relevanten Attribute, ihre Datentypen und Einschränkungen

aufgeführt. # Tabelle für Nutzer

<details open markdown="block">

<summary>- Nutzer/</summary>

<pre>

```

"nutzer_id"    INTEGER
PRIMARY KEY, "uni_id"
                INTEGER,
"name"        TEXT NOT NULL,
"email"       TEXT UNIQUE NOT NULL,
"passwort"
                TEXT
NOT NULL, "alter"
                INTEG
ER,
"erstellt_am" DATE DEFAULT CURRENT_DATE,
FOREIGN KEY ("uni_id") REFERENCES universität("uni_id")

```

</pre>

</details>

# Tabelle für Universitäten

<details open markdown="block">

<summary>- Universität</summary>

<pre>

```

"uni_id"    INTEGER
PRIMARY KEY, "name"
                TEXT
NOT NULL,
"kurzname"
me"          TEXT, "standort"    TEXT

```

4/6

</pre>

</details>

# Tabelle für Persönlichkeitsergebnisse

<details open markdown="block">

<summary>- Persönlichkeitsergebnis</summary>

<pre>

```
"ergebnis_id" INTEGER
PRIMARY KEY, "nutzer_id"
            INTEGER,
"vec_ei"     FLOAT,
"vec_sn"     FLOAT,
"vec_tf"     FLOAT,
"vec_jp"
            FL
OAT,
"mbti_typ"
            T
EXT,
"abgeschlosse
n_am" DATE,
FOREIGN KEY ("nutzer_id") REFERENCES nutzer("nutzer_id")
```

</pre>

</details>

# Tabelle für Swipes

<details open markdown="block">

<summary>- Swipe</summary>

<pre>

```
"swipe_id"      INTEGER PRIMARY KEY,
"swiper_id"     INTEGER, "swiped_id"    INTEGER,
"swipe_typ"     TEXT, -- "like"
oder "dislike" "geswiped_am"
            DATE,
"match_generiert" BOOLEAN DEFAULT FALSE,
FOREIGN KEY ("swiper_id") REFERENCES
nutzer("nutzer_id"), FOREIGN KEY ("swiped_id")
REFERENCES nutzer("nutzer_id")
```

</pre>

</details>

# Tabelle für Matches

```
<details open markdown="block">
```

```
<summary>- Match/</summary>
```

```
<pre>
```

```
"match_id"    INTEGER PRIMARY KEY,
```

5/6

Wenn du möchtest, kann ich dir das auch als `.md`-Datei für GitHub Pages generieren. Sag einfach Bescheid!



