

# Semesterarbeit

# CRUD Applikation

Simone Cogli und Konstantin Hirtz

TEKO Olten

# Inhaltsverzeichnis

Auftrag

Projektstruktur

Datenbankstruktur

Beschreibung der Seiten

Testbericht

Lessons Learnt

---

## Auftrag

---

Als Auftrag für das letzte Semester mussten wir eine Blog-Seite erstellen. Dafür sollte eine Flask-Applikation inklusive Datenbank verwendet werden. Die Blogseite soll eine vollständige CRUD Operation auf zwei Objekte sichern, Messaging mit zwei verschiedenen Typen sowie auch ein User Interface durch Bootstrap aufweisen.

Unser Team für diesen Auftrag bestand aus Konstantin Hirtz und mir (Simone Cogli).

---

## Projektstruktur

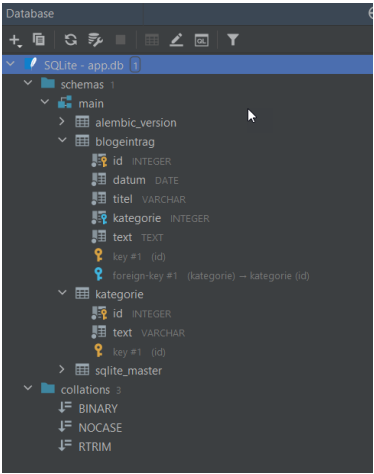
---

Für dieses Projekt waren wir leider nicht in der Lage, den Blueprint in unser Programm zu implementieren.

Das Programm haben wir nach unseren Möglichkeiten möglichst einfach aufgebaut. Hierfür haben wir einen Ordner mit all unseren Templates erstellt. Darin enthalten ist:

- |                            |   |
|----------------------------|---|
| - base.html                | → Vorlage für alle anderen Seiten (Bootstrap) |
| - category.html            | → Seite um die Kategorien aufzuzeigen         |
| - index.html               | → Startseite und Anzeige der Blogs            |
| - kategorie_erstellen.html | → Seite um neue Kategorien zu erstellen       |
| - neuerBlog.html           | → Seite um neue Blogs zu erstellen            |

Weitere Dateien:

App.db	
--------	--

## App.py

```
import ...

app = Flask(__name__)
app.config.from_object(Config)

db = SQLAlchemy()
migrate = Migrate()

db.init_app(app)
migrate.init_app(app, db)
from models import Blogeintrag, Kategorie

@app.route('/')
def index():
    blogeintraeg = Blogeintrag.query.all()
    return render_template('index.html', blogeintraeg=blogeintraeg)

@app.route('/category')
def category():
    data = Kategorie.query.all()
    return render_template('category.html', data=data)

@app.route('/remove/<int:id>')
def remove(id):
    db.session.remove()
    db.session.commit()
    kategorie = Kategorie.query.get_or_404(id)
    db.session.delete(kategorie)
    db.session.commit()
```

## Config.py

```
import os

basedir = os.path.abspath(os.path.dirname(__file__))

class Config():
    SECRETKEY = 'poierprtghzvqihpgfophôioi89ptphpihquz'
    SQLALCHEMY_DATABASE_URI = 'sqlite:///app.db'
    SQLALCHEMY_TRACK_MODIFICATIONS = False
```

## Forms.py

```
from wtforms import Form, StringField, DateField, IntegerField, TextAreaField
from wtforms.validators import DataRequired

class Blogform(Form):
    datum = DateField('datum', validators=[DataRequired()])
    titel = StringField('titel', validators=[DataRequired()])
    category = IntegerField('category', validators=[DataRequired()])
    eintrag = TextAreaField('eintrag', validators=[DataRequired()])

class Kategorieform(Form):
    category = StringField('category', validators=[DataRequired()])
```

## Models.py

```
from datetime import datetime

from app import db

class Blogeintrag(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    datum = db.Column(db.Date, nullable=False)
    titel = db.Column(db.String, nullable=False)
    kategorie = db.Column(db.ForeignKey('kategorie.id'), nullable=False)
    text = db.Column(db.Text, nullable=False)

class Kategorie(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    text = db.Column(db.String, nullable=False)
    blogeintrag = db.relationship('Blogeintrag', backref='Category', cascade='delete')
```

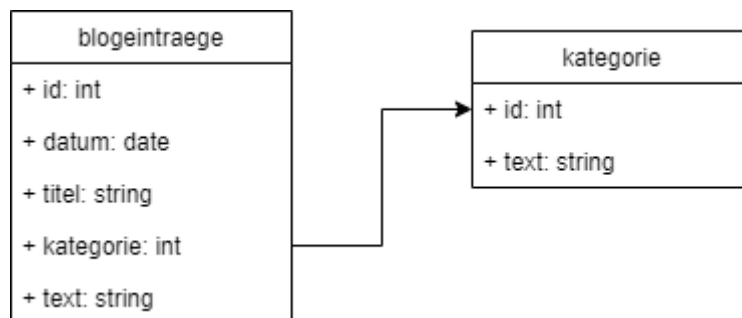
---

## Datenbankstruktur

---

Die Datenbankstruktur wollten wir so einfach wie möglich gestalten. Auf der einen Seite „blogeintraeg“ mit den Einträgen id, datum, titel, kategorie und text. Auf der anderen Seite haben wir „kategorie“ mit den Einträgen id und text.

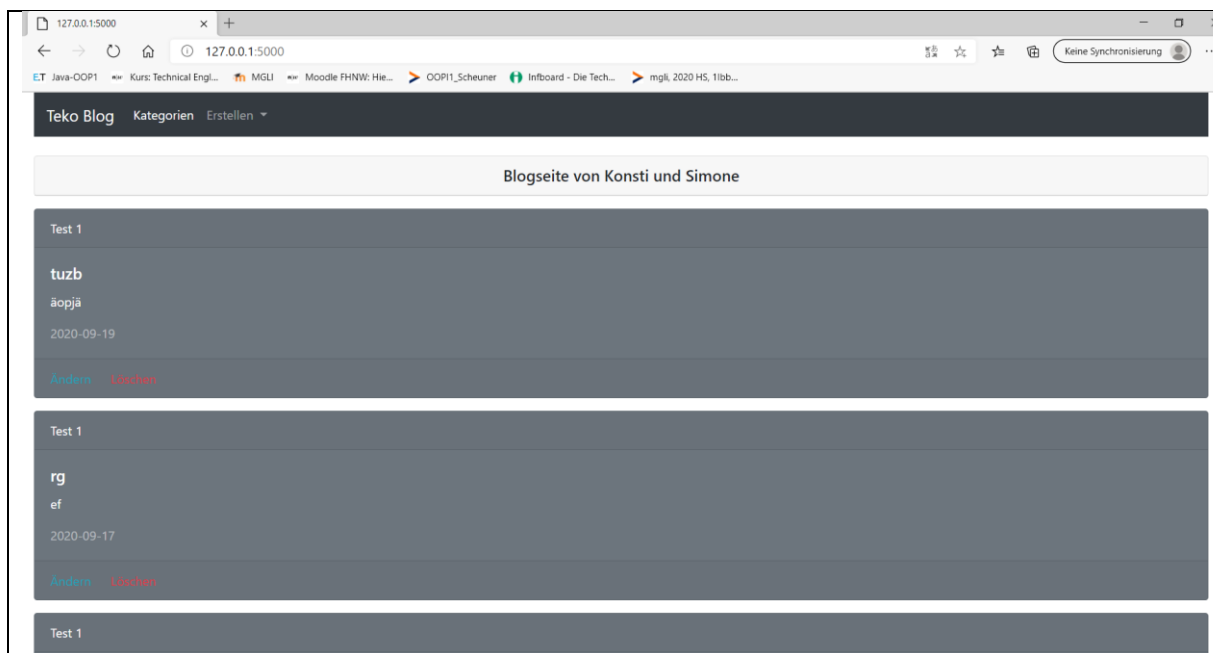
Die kategorie von „blogeintraege“ ist hier von der id der „kategorie abhängig.



---

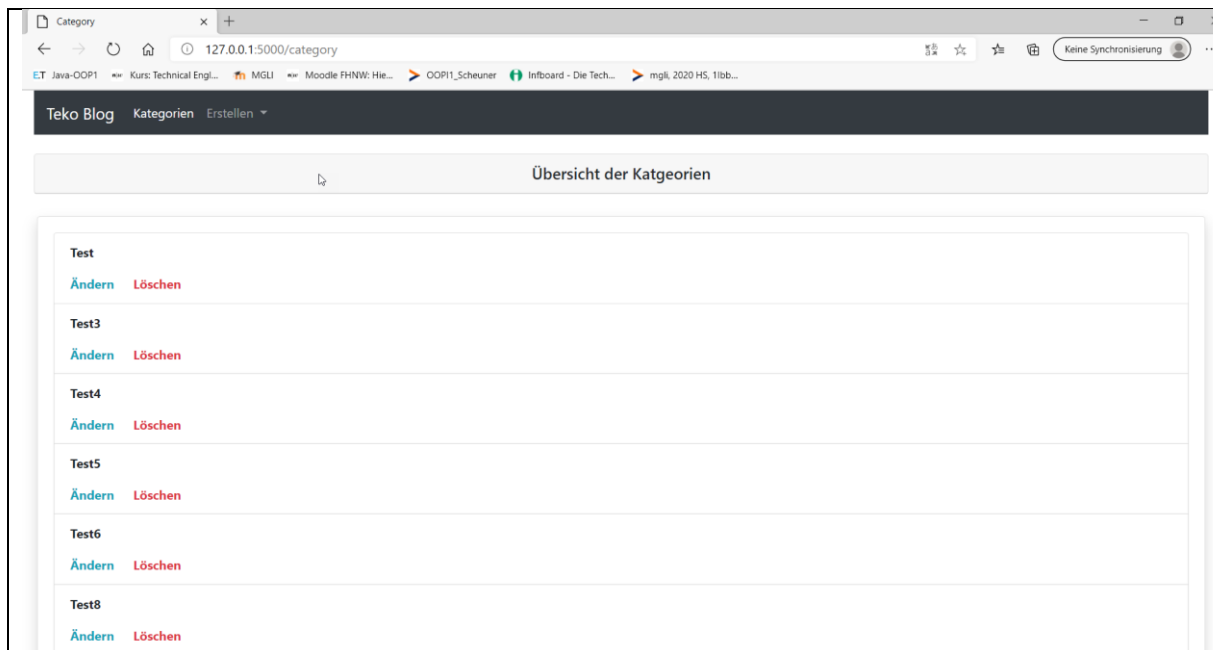
## Beschreibung der Seiten

---

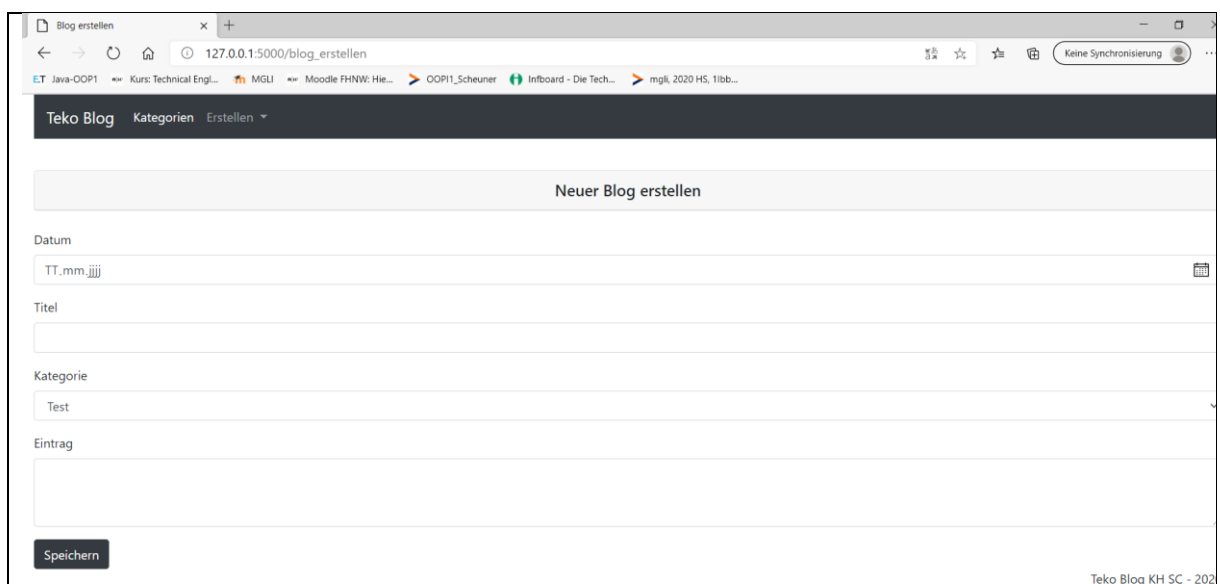


Dies ist die Startseite und zeitgleich auch die Blog-Anzeige-Seite. Hier landet man beim Öffnen der Seite und kann direkt alle Blogbeiträge sehen. Von hier aus sollte man sich durch die Blogs lesen und sie eventuell bearbeiten oder sogar löschen können.

Außerdem kann man mittels Navigationsbar auf die zweite Seite «Kategorien» wechseln oder das Drop-Down-Menü öffnen um eine neue Kategorie oder einen neuen Blog zu erstellen.



Hier werden alle erstellten Kategorien angezeigt. Diese sollten zu Verfügung stehen um geändert und gelöscht zu werden.  
Ausserdem kann man über die Navigationsbar je nach Bedarf zwischen den einzelnen Seiten switchen.



Auf diese Seite gelangt man durch das Öffnen des Drop-Down-Menüs und anschliessendem Betätigen der Auswahl «Blog erstellen». Hier kann man ein Datum eingeben, dann einen Titel vergeben und anschliessend die Kategorie wählen, in der der Post gemacht werden soll. Zum Schluss kann man noch im Feld «Eintrag» seinen Text schreiben. Nachdem all diese Felder ausgefüllt wurden kann man auf «Speichern» klicken und der Blog wird in die Datenbank geschrieben und auf der Homepage angezeigt.

Kategorie erstellen

127.0.0.1:5000/kategorie\_erstellen

Teko Blog Kategorien Erstellen

Neue Kategorie erstellen

Kategorie

Speichern

Teko Blog KH SC - 202

Auf dieser Seite kann man schliesslich noch die Kategorien erstellen. Diese werden können dann auf über den Link «Kategorien» bearbeitet und gelöscht werden.

---

## Testbericht

---

Funktion	Funktionsfähigkeit	Begründung falls nein
Homepage öffnen	Ja	
Blogs anschauen	Ja	
Blogs löschen	Ja	
Blogs bearbeiten	Nein	<p>Wir haben es nach unseren Möglichkeiten versucht, dieses Problem zu lösen. Doch auch nach mehreren Stunden Arbeit konnten wir das Problem leider nicht lösen.</p> <p><code>AttributeError: 'Kategorie' object has no attribute 'kategorie'</code></p>
Kategorie Seite öffnen	Ja	
Kategorien ansehen	Ja	
Kategorien löschen	Ja	
Kategorien bearbeiten	Nein	<p>Gleich wie bei den Blogs, konnten wir dieses Problem nicht lösen. Auch hier, wie auch bei den Kategorien haben wir viel Zeit verloren. Das Problem hängt mit dem der Blogs zusammen.</p> <p><code>AttributeError: 'Kategorie' object has no attribute 'kategorie'</code></p>
«Erstellen» öffnen	Ja	
Blog-Erstellungsseite öffnen	Ja	
Datum Eingabe	Ja	
Titel Eingabe	Ja	
Kategorie Auswahl	Ja	
Eintrag Eingeben	Ja	
Blog speichern	Ja	
Kategorie-Erstellungsseite öffnen	Ja	
Neue Kategorie erfassen	Ja	
Neue Kategorie speichern	Ja	



---

## Lessons Learnt

---

### Lessons Learned Konstantin Hirtz

Im vergangenen Semester wurde uns der Auftrag erteilt. Eine Python / Flask Applikation zu erweitern. Als Ausgangslage erhielten wir den Bierblog von unserem Dozenten. Die Herausforderung war, die bestehende Ausgangslage zu verstehen und noch weitere Funktionen hinzuzufügen. Interpretieren und analysieren (verstehen was der Code macht) liegt mir sehr. Jedoch fällt mir das debuggen von Code schwer, da ich einfach zu wenig Zeit damit verbracht habe. Die Zeitinvestition welche man tätigen muss, um die Zusammenhänge zu verstehen sind für mich enorm und ich konnte den Aufwand, gegenüber meinen anderen Verpflichtungen, nicht rechtfertigen. Das Fach und der Auftrag würde mir, unter anderen Umständen sehr zusagen.

Was habe ich dabei gelernt?

Ich habe privat nie etwas mit Flask am Hut gehabt und für mich war vieles Neuland.

Wir haben in diesem Projekt erlernt den Flask Code zu verstehen und die, im Unterricht erlernten Methoden wie Löschen / Erstellen / Verändern von Einträgen wie auch die Anbindung einer Flask-App an eine Datenbank, anzuwenden.

Was hätte ich anders gemacht?

Wir haben, dadurch dass Simone wie auch ich an einer weiteren Schule bereits eine Zweitbelastung hatten und die Diplomarbeit der Teko auch in diesem Zeitrahmen stattgefunden hat, das Projekt vernachlässigt und sind dadurch in Zeitnot geraten. Das Zeitmanagement von uns, hat so gut wie keine Parallelen aufgezeigt und somit konnten wir erst nach der Diplomarbeit, effektiv an dem Projekt arbeiten. Wir hätten uns hier die Arbeit aufteilen sollen oder unabhängig voneinander am Projekt arbeiten sollen.

### Lessons Learned Simone Cogli

Im Verlauf des Projektes, merkte ich, wie man immer mehr mit dem ganzen System vertraut ist, je mehr man sich auch damit beschäftigt. Dies war für mich und Konstantin eine grosse Herausforderung. Aufgrund meiner begrenzten Zeit, konnte ich zum Anfang des Projektes nicht alle Ressourcen verwenden, wie ich es mir gewünscht hätte. Wäre es mir möglich gewesen, von Anfang an immer mit dem Projekt Schritt zu halten, dann wären wir jetzt am Schluss sicherlich nicht in solch ein grossen Stress geraten. Ausserdem bin ich mir sicher, dass wir ein besseres Produkt hätten liefern können. Eine zweite Schule, der Abschluss der HF sowie die Diplomarbeit waren zusammen einfach zu viel, um überall die höchste Qualität zu gewährleisten. Aus diesem Grund haben wir auf eine Art den Zug fast verpasst. Trotz allem, konnten mehr oder weniger ein Programm auf die Beine stellen. Es gibt zwar einige Mangel punkt, die aber meiner Meinung nach nicht das positive der Arbeit bedecken.

Ich konnte während diesem Semester jedoch vieles neues lernen, da es auch ein ziemlichen Unterschied zum letzten Semester war und ich noch nie etwas zuvor in dieser Richtung zu tun gehabt habe. Vor allem im Vergleich zu TigerJython.

An dieser Stelle möchten wir uns auch noch bei denjenigen bedanken, welche uns bei unserem Projekt eine Unterstützung bieten konnten.