CRUD Webanwendung mittels express

Autor: Mario Fentler
Datum: 08.12.2018

Für die Übung wurde ein Webserver mit Express erstellt. Auf dem kann man über ein Formular Schueler in eine MongoDb einfügen. Required packages

```
npm init
npm install mongodb --save
npm install express --save express
npm install body-parser --save
```

In die Datei **index.ejs** (ejs steht für extended JavaScript) wird der HTML Code der Webseite rein geschrieben. Auf der Webseite wird für die schönere Visualisierung das Bootstrap Framework verwendet.
Es kann auch JavaScript Code, der **auf dem Server ausgeführt** werden soll (ähnlich wie Ajax) in dieses File eingefügt werden. Das macht man so:

```
<% for (var i = 0; i < schueler.length; ++i) { %>
Mit den <%= %>
```

Aufgabendurchführung

Als erster Schritt werden im js File Konstanten erstellt.

```
const express = require('express')
const MongoClient = require('mongodb').MongoClient
const mongoDB = require('mongodb')
const app = express()

//Damit ich die Formulare als JSON bekomme
const bodyParser = require('body-parser')
app.use(bodyParser.urlencoded({extended:false}))
```

MongoDB - mlab

Als Datenbank wird mongoDB verwendet, welches über die Webseite von m-lab.com erreichbar ist. Dort erstellt man sich eine neue Datenbank und eine Collection.

Anschließend kann man sich im Server über den MongoClient damit verbinden.

```
MongoClient.connect('mongodb://databaseUser:avHg6Ny4aAwXikpzde@ds113454.mlab.com:1
3454/5chit_insy', (err,databaseConnection)=>{
if (err) {
    console.log('Fehler bei der Verbindung zur Datenbank')
    console.log(err)
}else{
    db = databaseConnection.db('5chit_insy')
    app.listen(8080, ()=>{
        console.log('Server laeuft auf 127.0.01:8080')
    })
}
```

}) Sofern keine Fehler beim Verbinden mit mlab auftreten, kann der Server mit **app.listen(<port>)** gestartet werden.

Routing

Im Server werden einzelnen Routen verschiedene Methoden zugewiesen. Die zum auslesen aller Schüler, die standardmäßig über '/' aufgerufen wird, könnte so aussehen:

```
app.get('/', (req,res)=>{
    db.collection('schueler').find().toArray((err,result)=>{
        if(err){
            console.log(err)
        }else{
            res.render('index.ejs', {schueler:result})
        }
    })
})
```

Dabei wird über **db.collection(<name>)..find().toArray()** auf die Dokumente in der Datenbank zugegriffen und diese dann an die Webseite weitergeleitet.
toArray() wird verwendet, da es sich hier um mehere Ergebnisse handeln kann.

Anzeige auf der Webseite

Die Webseite, in dem Fall index.ejs und edit.ejs(mehr zu dieser später) enthalten den HTML-Code der Webseite. Neben Formularen werden dort auch JavaScript funktionen eingebunden.

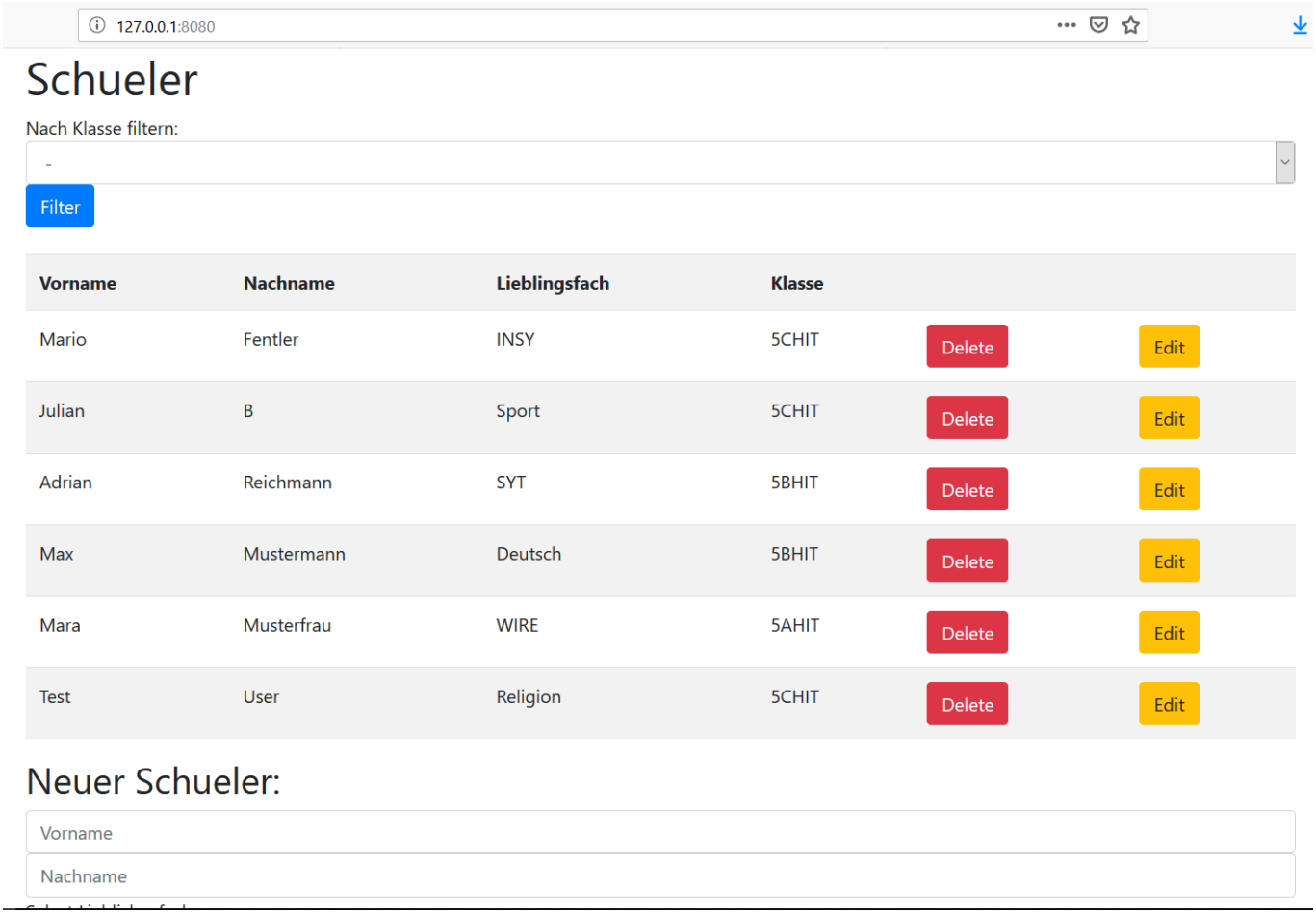
```
//index.ejs
<script>
    $("#filterClass").change(function() {
    var action = $(this).val();
    $("#filterClassForm").attr("action", "/search/" + action);
    });
</script>
```

Dieses Script wird allerdings nur gebraucht um die action vom Formular auf die ausgewählte Listen-Option umzuschreiben. Diese Funktion wird für die Suchanforderung gerbaucht.

Funktionen der Webseite

- Schüler auflisten

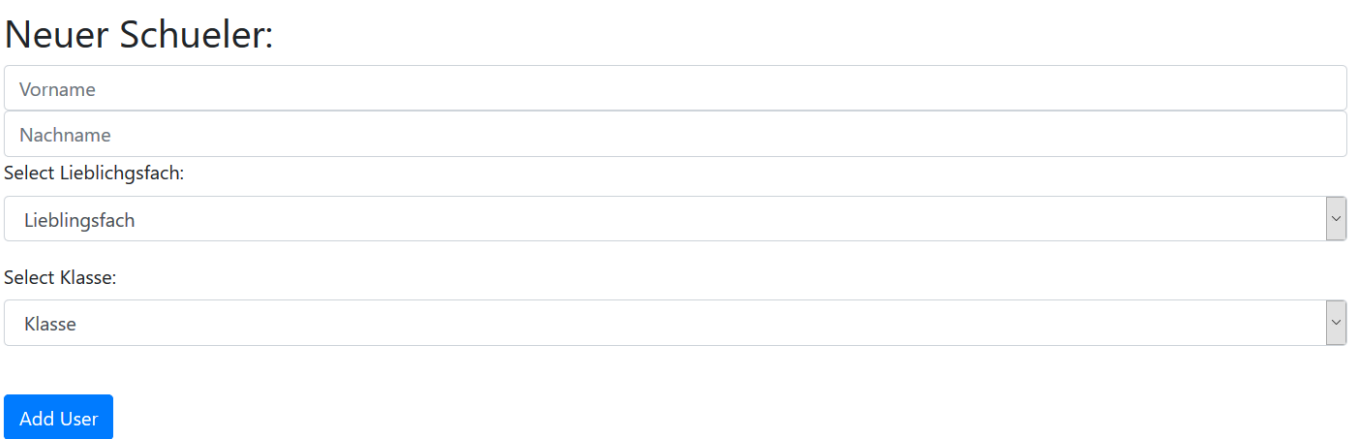
Wie die Schüler aus der Datenbank ausgelesen und ausgegeben werden wurde vorher schon beschrieben.



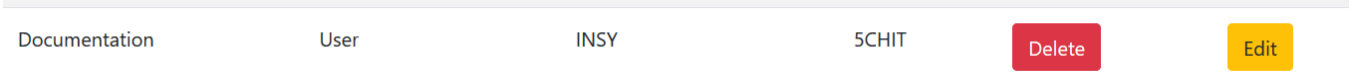
- Schüler erstellen

Neue Benutzer können über ein Formular auf der Webseite angelegt werden. Nachdem der User dieses Formular submitted wird der Inhalt in einem JSON-Format an den Server weitergeleitet. Mit der Methode **db.collection(<name>).insertOne()** können neue Einträge in die Datenbank eingefügt werden.

```
app.post('/new', (req,res)=>{
    db.collection('schueler').insertOne(req.body, (err, result) =>
        {...}
    })
})
```



Nachdem der neue Schüler erstellt wurde ist er in der Liste auffindbar.



- Schueler editieren

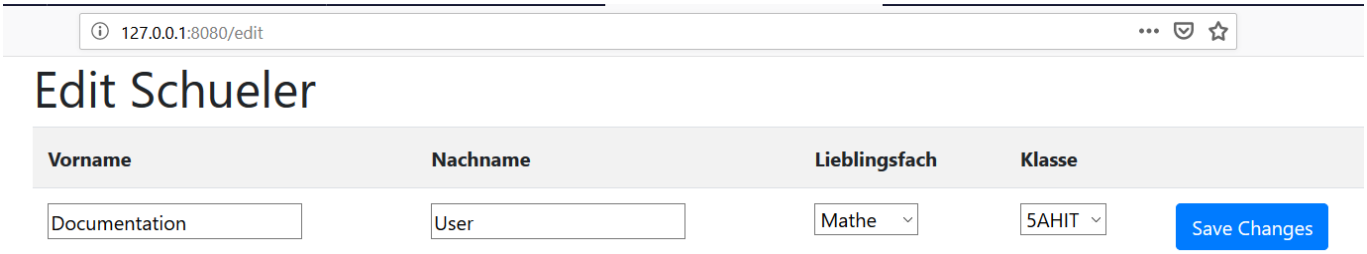
Man kann die Werte eines Schuelers auch ändern. Dazu klickt man auf den *EDIT* Button. Dieser beinhaltet die id vom User, mit der der Server dann die restlichen Werte von der MongoDB abfragen kann und auf einer neuen Seite (edit.ejs) als editierbare Textfelder anzeigen kann.

```
<form class="form" action="/edit" method="POST">
    <td>
    <input type="hidden" id="id" name="id" value="<%= schueler[i]._id %>" />
    <input class="btn btn-warning" type="submit" value="Edit"/>
    </td>
</form>

app.post('/edit', (req,res)=>{
    db.collection('schueler').find({_id:
mongoDB.ObjectID(req.body.id)}).toArray((err,result)=>{
        if(err){
            console.log(err)
        }else{
            res.render('edit.ejs', {schueler:result[0]})
        }
    })
})
```

Mit der Methode **db.collection(<name>).res.render()** kann man im Code auf eine neue Seite weiterleiten.

Mit der Methode **db.collection(<name>).updateOne()** können Dokumente in der MongoDB geändert werden. Dazu muss man als "fixen Wert" die ID mitgeben. Zu der ist zu sagen, dass die id eine **mongoDB.ObjectID** sein muss.

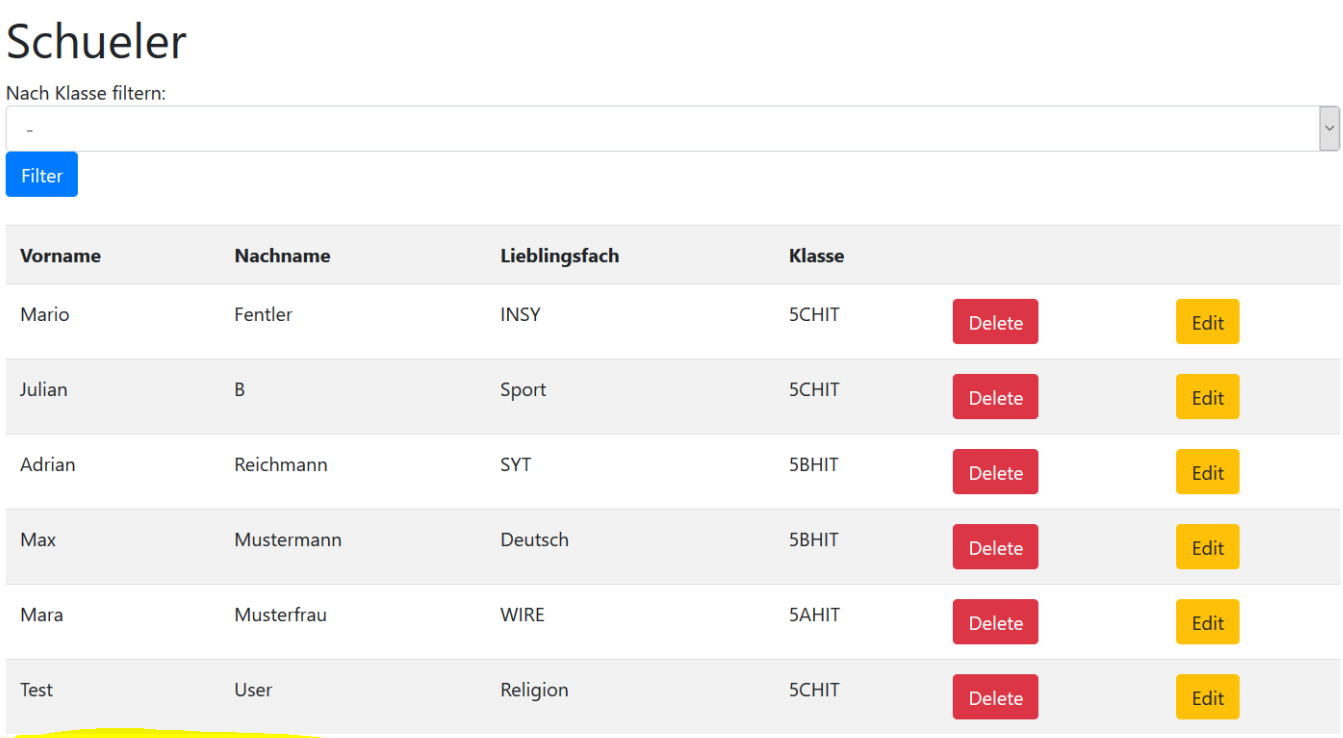
```
app.post('/edit/:status', (req,res)=>{

    db.collection('schueler').updateOne({_id: mongoDB.ObjectID(req.body.id)},
    {$set: {"vorname":req.body.vorname,"nachname":req.body.nachname,
    "lieblingsfach":req.body.lieblingsfach,
    "klasse":req.body.klasse}}, (err,result)=>{...}
    })
})
```



Sofern man dort dann auf den *SAVE Changes* Button drückt wird man zurück auf die index.ejs Seite geleitet. Dort sind nun die getätigten Änderungen ersichtlich.

- Delete Schueler

Zu guter letzt kann ein Schueler auch noch über den Button delete aus der MongoDB geloescht werden. Dazu wird die Methode **db.collection(<name>).deleteOne()** verwendet.



Neuer Schueler:

Wie man hier sehen kann wurde der Schueler gelöscht.

- Nach Klasse filtern

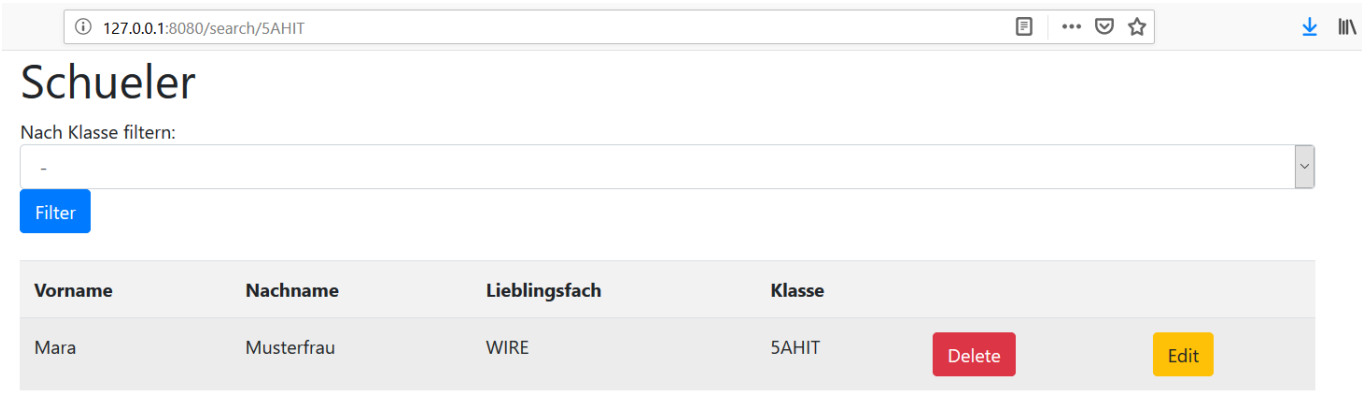
Als letzter Schritt wurde eine Filtermöglichkeit eingefügt. Man kann über eine Liste auswählen welche Schüler (Klasse) man sehen möchte.
Sofern nicht "Alle" als Option ausgewählt wurde, werden nur noch die Schüler aus der richtigen Klassen angezeigt.

Für die Funktionalität des Filtern wird eine post Methode verwendet, da sie vom Formular auf der Webseite über POST aufgerufen wird. In dieser Methode wird gefiltert ob alle Schüler aus der MongoDB ausgelesen werden sollen oder nur die mit einer bestimmten Klasse.

```
app.post('/search/:class', (req,res) => {
    if(req.body.filterclass == "Alle"){
        db.collection('schueler').find().toArray((err,result)=>{
            if(err){
                console.log(err)
            }else{
                res.render('index.ejs', {schueler:result})
            }
        })
    }else{

db.collection('schueler').find({"klasse":req.body.filterclass}).toArray((err,resul
t) => {
            if(err){
                console.log(err)
            }else{
                res.render('index.ejs', {schueler:result})
```

```
            }
        })
    }
})
```

Das Resultat, wenn man nach der Klasse *5AHIT* filtert sieht dann so aus:



Deployen

Um das Programm zu starten muss man Node.js installiert haben. Danach startet man den Server mit folgendem Befehl:

```
node schueler.js
```

Anschließend ist der Server über **127.0.0.1:8080** erreichbar.

Fehler die während der Übung aufgetreten sind

- #1

Ich habe viel Zeit damit verbracht zu eruieren wieso die JQuery Funktion nicht funktioniert.
-> Man muss natürlich JQuery laden damit es funktioniert (face-palm)

```
\<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">\
</script>
```

- #2

Damit man ein Schueler Objekt in der MongoDB über die ID finden kann muss diese ID, die man vom JSON-Request bekommt, in ein **mongoDB.ObjectID** Objekt umgewandelt werden.

Sources

https://stackoverflow.com/questions/4932928/remove-by-id-in-mongodb-console
https://stackoverflow.com/questions/736590/add-new-attribute-element-to-json-object-using-javascript

https://www.w3schools.com/tags/att_input_type_hidden.asp
https://docs.mongodb.com/manual/reference/method/db.collection.updateOne/
https://stackoverflow.com/questions/42396025/express-cannot-post-quotes
https://mongodb.github.io/node-mongodb-native/markdown-docs/queries.html
https://www.w3schools.com/jquery/jquery_get_started.asp