Upload.routes.js

const express = require('express');

const router = express.Router();

const upload = require('../middleware/upload');

const Post = require('../models/posts');

router.post('/', upload.single('file'), async(req, res) => {

    if(req.file === undefined)

    {

        return res.send({

            "message": "no file selected"

        })

    } else {

        const newPost = new Post({

            title: req.body.title,

            location: req.body.location,

            image\_id: req.file.filename,

        })

        await newPost.save();

        sendNotification();

        return res.send(newPost);

    }

})

 /\* router.post('/', upload.single('file'), (req, res) => {

    if (req.file === undefined) {

        return res.send({

            "message": "no file selected"

        });

    } else {

        console.log('req.file', req.file);

        const imgUrl = `http://localhost:3001/download/${req.file.filename}`;

        return res.status(201).send({

            url: imgUrl

        });

    }

})

 \*/

module.exports = router;

download.routes.js

const express = require('express');

const router = express.Router();

module.exports = router;

posts.routes.js

const express = require('express');

const router = express.Router();

const { collection } = require('../configure/db')

const  ObjectId = require('mongodb').ObjectId

// eine GET-Anfrage

router.get('/', async(req, res) => {

   const allPosts = await collection.find().toArray();

    res.status(200);

    res.send(allPosts);

    // res.send({ message: "Hello FIW!" });

});

//  CREATE  one new post

router.post('/', async(req, res) => {

    try {

        const newPost = {

            title: req.body.title,

            location: req.body.location,

            image\_id: req.body.image\_id

        }

        const result = await collection.insertOne(newPost);

        res.status(201);

        res.send(result);

    } catch {

        res.status(404);

        res.send({

            error: "Post did not send!"

        });

    }

});

// GET one post by id

router.get('/:id', async(req, res) => {

    try {

        const id\_obj = new ObjectId(req.params.id);

        const post = await collection.find( {\_id: id\_obj } ).toArray();

        console.log('post', req.params.id)

        res.status(202);

        res.send(post);

    } catch {

        res.status(404);

        res.send({

            error: "Post does not exist!"

        });

    }

});

// PATCH (update) one post

router.patch('/:id', async(req, res) => {

    try {

        const id\_obj = new ObjectId(req.params.id);

        const post = await collection.findOne({ \_id: id\_obj })

        if (req.body.title) {

            post.title = req.body.title

        }

        if (req.body.location) {

            post.location = req.body.location

        }

        if (req.body.image\_id) {

            post.image\_id = req.body.image\_id

        }

await collection.updateOne({ \_id: id\_obj }, { $set: post });

res.send(post)

} catch { res.status(404), res.send({ error: "Post does not exist!" }) }

});

// DELETE one post via id

router.delete('/:id', async(req, res) => {

    try {

        const id\_obj = new ObjectId(req.params.id);

        const post = await collection.deleteOne({ \_id: id\_obj })

        console.log('post', post)

        if(post.deletedCount === 1) {

            res.status(204)

            res.send( { message: "deleted" })

        } else {

            res.status(404)

            res.send({ error: "Post does not exist!" })

        }

    } catch {

        res.status(404)

        res.send({ error: "something wrong" })

    }

});

module.exports = router;

db.js

const { MongoClient } = require('mongodb');

require('dotenv').config();

const client = new MongoClient(process.env.DB\_CONNECTION);

const dbconnection = client.connect();

const database = client.db(process.env.DB\_NAME);

const collection = database.collection(process.env.COLLECTION);

console.log(`Connected to DB ... `);

module.exports.client = client;

module.exports.dbconnection = dbconnection;

module.exports.database = database;

module.exports.collection = collection;

server.js

const express = require('express');

const postRoutes = require('./routes/posts.routes');

const uploadRoutes = require('./routes/upload.routes');

const downloadRoutes = require('./routes/download.routes');

const cors = require('cors')

const mongoose = require('mongoose');

require('dotenv').config();

const app = express();

const PORT = 3001;

app.use(express.json());

// enable cors for all requests

app.use(cors());

app.use('/posts', postRoutes);

app.use('/upload', uploadRoutes);

app.use('/download', downloadRoutes);

app.listen(PORT, (error) => {

    if (error) {

        console.log(error);

    } else {

        console.log(`server running on http://localhost:${PORT}`);

    }

});

Feed.js

var shareImageButton = document.querySelector('#share-image-button');

var createPostArea = document.querySelector('#create-post');

var closeCreatePostModalButton = document.querySelector('#close-create-post-modal-btn');

let sharedMomentsArea = document.querySelector('#shared-moments');

let networkDataReceived = false;

function openCreatePostModal() {

  createPostArea.style.display = 'block';

}

function closeCreatePostModal() {

  createPostArea.style.display = 'none';

}

shareImageButton.addEventListener('click', openCreatePostModal);

closeCreatePostModalButton.addEventListener('click', closeCreatePostModal);

function createCard(card) {

  let cardWrapper = document.createElement('div');

  cardWrapper.className = 'shared-moment-card mdl-card mdl-shadow--2dp';

  let cardTitle = document.createElement('div');

  cardTitle.className = 'mdl-card\_\_title';

  let image = new Image();

  image.src = card.image.id;

  cardTitle.style.backgroundImage = 'url('+ image.src +')';

  cardTitle.style.backgroundSize = 'contain';

  cardTitle.style.backgroundRepeat = 'no-repeat';

  cardTitle.style.backgroundPosition = 'center';

  cardTitle.style.backgroundColor = '#FBF4ED'

  cardTitle.style.height = '180px';

  cardWrapper.appendChild(cardTitle);

  let cardTitleTextElement = document.createElement('h2');

  cardTitleTextElement.className = 'mdl-card\_\_title-text';

  cardTitleTextElement.textContent = card.title;

  cardTitle.appendChild(cardTitleTextElement);

  let cardSupportingText = document.createElement('div');

  cardSupportingText.className = 'mdl-card\_\_supporting-text';

  cardSupportingText.textContent = card.location;

  cardSupportingText.style.textAlign = 'center';

  cardWrapper.appendChild(cardSupportingText);

  componentHandler.upgradeElement(cardWrapper);

  sharedMomentsArea.appendChild(cardWrapper);

}

function updateUI(data) {

  for(let card of data)

  {

     createCard(card);

  }

}

fetch('http://localhost:3001/posts')

    .then((res) => {

        return res.json();

    })

    .then((data) => {

      networkDataReceived = true;

      console.log('From backend ...', data);

        updateUI(data);

    });

  if('indexedDB' in window) {

      readAllData('posts')

          .then( data => {

              if(!networkDataReceived) {

                  console.log('From cache ...', data);

                  updateUI(data);

              }

          })

  }

Sw.js

importScripts('/src/js/idb.js');

//importScripts('/src/js/db.js');

/\* importScripts(

    'https://storage.googleapis.com/workbox-cdn/releases/6.4.1/workbox-sw.js'

  );

  workbox.routing.registerRoute(

      ({request}) => request.destination === 'image',

      new workbox.strategies.NetworkFirst()     // NetworkFirst() vs CacheFirst()

  ) \*/

  const CACHE\_VERSION = 6;

  const CURRENT\_STATIC\_CACHE = 'static-v' + CACHE\_VERSION;

  const CURRENT\_DYNAMIC\_CACHE = 'dynamic-v' + CACHE\_VERSION;

  const STATIC\_FILES = [

      '/',

      '/index.html',

      '/src/js/app.js',

      '/src/js/feed.js',

      '/src/js/material.min.js',

      '/src/js/idb.js',

      '/src/css/app.css',

      '/src/css/feed.css',

      '/src/images/htw.jpg',

      'https://fonts.googleapis.com/css?family=Roboto:400,700',

      'https://fonts.googleapis.com/icon?family=Material+Icons',

      'https://code.getmdl.io/1.3.0/material.blue\_grey-red.min.css'

  ]

  self.addEventListener('install', event => {

    console.log('service worker --> installing ...', event);

    event.waitUntil(

          caches.open(CURRENT\_STATIC\_CACHE)

            .then(cache => {

                console.log('Service-Worker-Cache erzeugt und offen');

                cache.add(STATIC\_FILES);

            })

    );

})

self.addEventListener('activate', function(event) {

    console.log('service worker --> activating ...', event);

    event.waitUntil(

        caches.keys()

            .then( keyList => {

                return Promise.all(keyList.map( key => {

                    if(key !== CURRENT\_STATIC\_CACHE && key !== CURRENT\_DYNAMIC\_CACHE) {

                        console.log('service worker --> old cache removed :', key);

                        return caches.delete(key);

                    }

                }))

            })

    );

    return self.clients.claim();

})

self.addEventListener('fetch', event => {

    // check if request is made by chrome extensions or web page

    // if request is made for web page url must contains http.

    if (!(event.request.url.indexOf('http') === 0)) return; // skip the request if request is not made with http protocol

        const url = 'http://localhost:3001/posts';

        if(event.request.url.indexOf(url) >= 0) {

        event.respondWith(

            fetch(event.request)

                .then ( res => {

                    // hier Anfrage an http://localhost:3000/posts behandeln

                    const clonedResponse = res.clone();

                    clearAllData('posts')

                    .then( () => {

                        return clonedResponse.json()

                    })

                    .then( data => {

                            for(let key in data)

                            {

                                console.log('write data', data[key]);

                                writeData('posts', data[key]);

                            }

                    });

                    return res;

                })

            )

        } else {

    event.respondWith(

        caches.match(event.request)

            .then( response => {

                if(response) {

                    return response;

                } else {

                    return fetch(event.request)

                        .then( res => {     // nicht erneut response nehmen, haben wir schon

                            return caches.open(CURRENT\_DYNAMIC\_CACHE)      // neuer, weiterer Cache namens dynamic

                                .then( cache => {

                                    cache.put(event.request.url, res.clone());

                                    return res;

                                })

                        });

                }

            })

    );

}})