[OCR] Node Express server

server.js

```
const express = require('express');
const bodyParser = require('body-parser');
const request = require('request');
const fs = require('fs');
const path = require('path');
const multer = require('multer');

const ROOT_DIR = path.resolve(__dirname, '..'); //  Root
const FILE_UPLOAD_DIR = path.resolve(ROOT_DIR, 'tmp/files'); // OCR
const OCR_RESULT_DIR = path.resolve(ROOT_DIR, 'tmp/work'); // OCR

const OCR_API_URL = ''; // OCR API URL
const OCR_API_KEY = ''; // OCR API Key
const OCR_OPTIONS = {
    type: 'upload',
    coord: 'origin',
    skew: 'image',
    boxes_type: 'all',
    save_mask: 'true',
    textout: 'true',
    recog_form: 'true',
    extract_table: 'true'
};

const app = express();
const router = express.Router();
const uploader = multer({ dest: FILE_UPLOAD_DIR });

router.post('/request', uploader.single('file'), async (request, response) => {
    const page = (request.body || {}).page || 0;
    const file = request.file;
    const filePath = file.path;
    const originalName = file.originalname;
    const mimeType = file.mimetype;
    const extension = path.extname(originalName).split('.')[1] || mimeType.split('/')[1];
    const filePathWithExt = filePath + '.' + extension;

    fs.renameSync(filePath, filePathWithExt);
    try {
        const ocrResult = await requestOCR(filePathWithExt, page);
        const imagePath = await downloadImage(ocrResult.result.masked_image);
        response.end(JSON.stringify({ result: ocrResult.result, imagePath }));
    } catch (error) {
        response.status(error.status).send(error.message);
    }
});

/**
 * OCR   .
 * @param {string} imgPath
 * @param {number} page
 * @returns
 */
function requestOCR(imgPath, page) {
    return new Promise((resolve, reject) => {
        const options = {
            url: `${OCR_API_URL}/ocr`,
            formData: {
                api_key: OCR_API_KEY,
                image: fs.createReadStream(imgPath),
                page_index: page,
                ...OCR_OPTIONS
            }
        };
```

```
        request.post(options).on('response', response => {
            const statusCode = response.statusCode;
            const statusMessage = response.statusMessage;
            if (statusCode === 200) {
                console.log('OCR ');
                let body = [];
                response.on('error', (err) => {
                    throw err;
                }).on('data', (chunk) => {
                    body.push(chunk);
                }).on('end', () => {
                    resolve(JSON.parse(Buffer.concat(body).toString()));
                });
            } else {
                console.log('OCR ', statusCode);
                const error = new Error(statusMessage);
                error.status = statusCode;
                reject(error);
            }
        });
    });
}

/**
 * OCR    .
 * @param {string} fileName
 * @returns
 */
function downloadImage(fileName) {
    return new Promise((resolve, reject) => {
        const options = {
            url: `${OCR_API_URL}/out/${fileName}`,
            formData: {
                api_key: OCR_API_KEY,
            }
        };

        const imagePath = path.resolve(OCR_RESULT_DIR, fileName);
        request.post(options).on('response', response => {
            const statusCode = response.statusCode;
            const statusMessage = response.statusMessage;
            if (statusCode === 200) {
                console.log('OCR image download ');

                let imageData = Buffer.from([]);
                response.on('data', (chunk) => {
                    imageData = Buffer.concat([imageData, chunk]);
                }).on('end', () => {
                    const relativeImagePath = '/' + path.relative(ROOT_DIR, imagePath).replace(/\\/g, '/');
                    fs.writeFileSync(imagePath, imageData); //
                    resolve(relativeImagePath);
                });
            } else {
                console.log('OCR image download ', statusCode);
                const error = new Error(statusMessage);
                error.status = statusCode;
                reject(error);
            }
        });
    });
}

app.use(bodyParser.json());
app.use('/', router);
app.listen(8080);
```