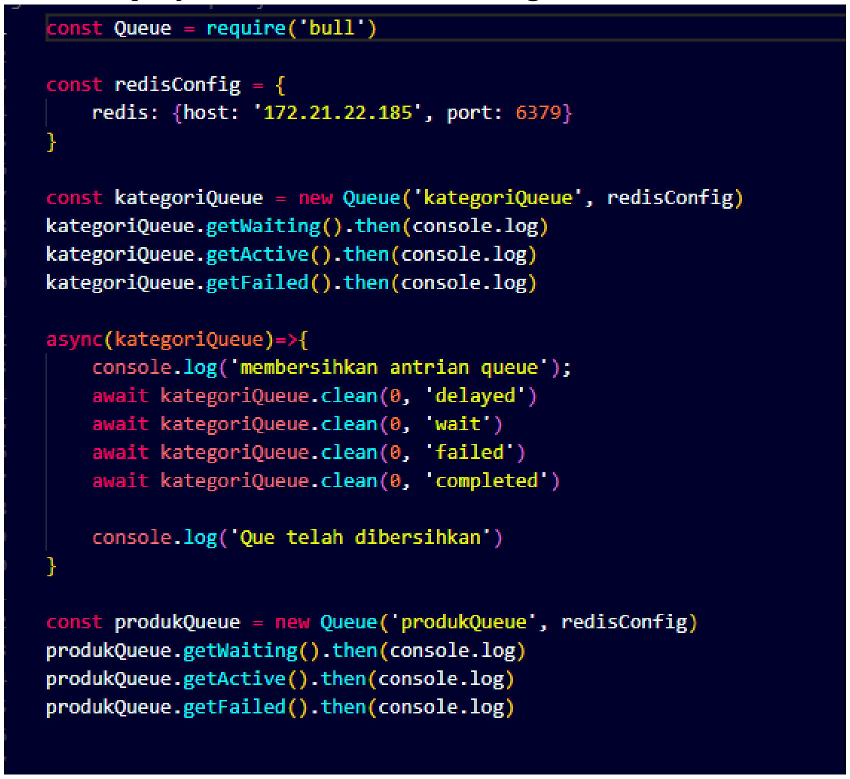
LAPORAN PRAKTIKUM PEMROGRAMAN FRAMEWORK

Rahadyan Danang Susetyo Pranawa

3123522018

1. Instalasi bull ioredis

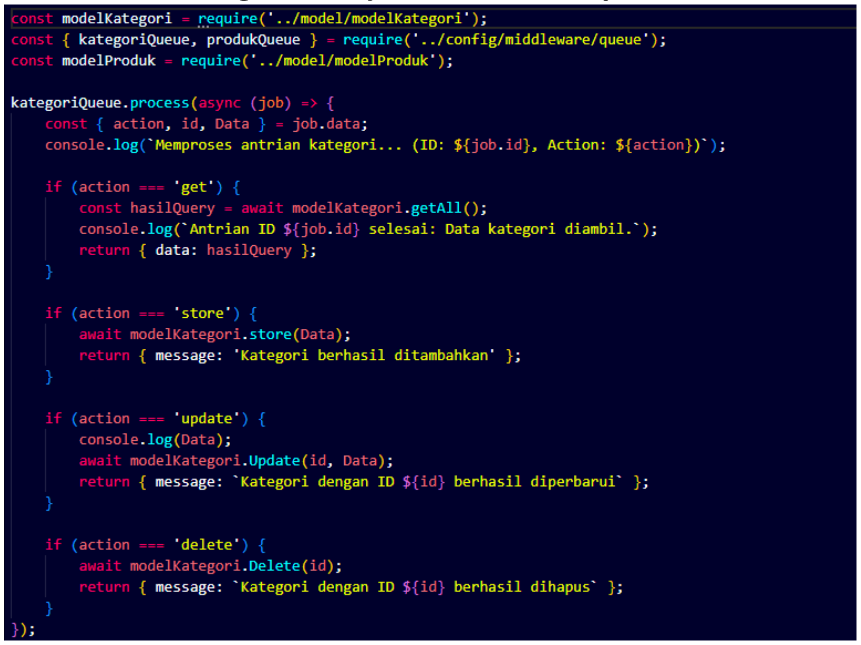
2. Buat file que.js dala,m middleware/config



```
const Queue = require('bull')

const redisConfig = {
    redis: {host: '172.21.22.185', port: 6379}
}

const kategoriQueue = new Queue('kategoriQueue', redisConfig)
kategoriQueue.getWaiting().then(console.log)
kategoriQueue.getActive().then(console.log)
kategoriQueue.getFailed().then(console.log)

async(kategoriQueue)=>{
    console.log('membersihkan antrian queue');
    await kategoriQueue.clean(0, 'delayed')
    await kategoriQueue.clean(0, 'wait')
    await kategoriQueue.clean(0, 'failed')
    await kategoriQueue.clean(0, 'completed')

    console.log('Que telah dibersihkan')
}

const produkQueue = new Queue('produkQueue', redisConfig)
produkQueue.getWaiting().then(console.log)
produkQueue.getActive().then(console.log)
produkQueue.getFailed().then(console.log)
```

que js berfungsi untuk konfigurasi redis

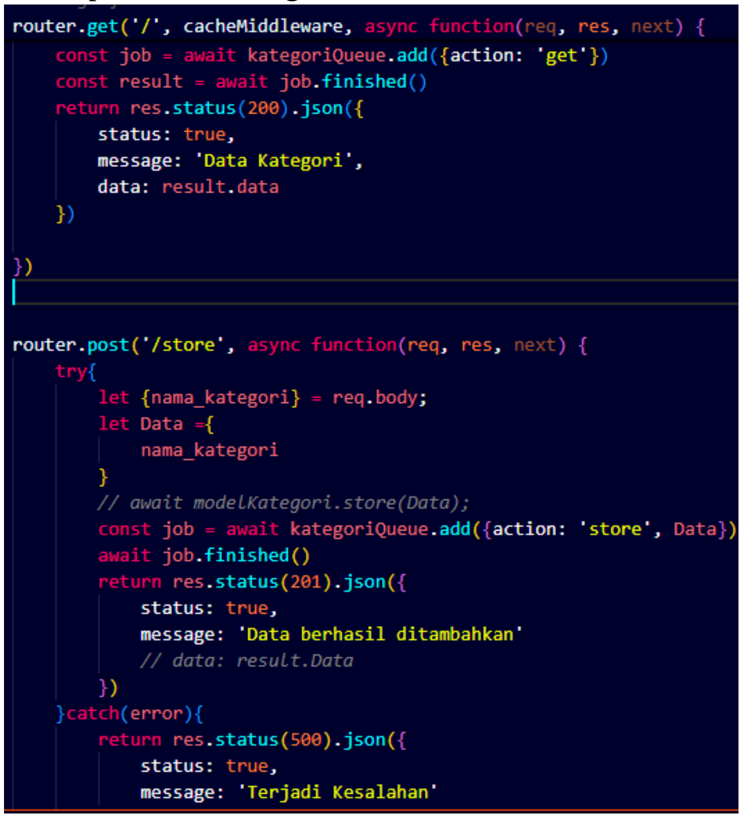
3. Buat direktori dengan nama jobs dan didalamnya buat file bernama workers



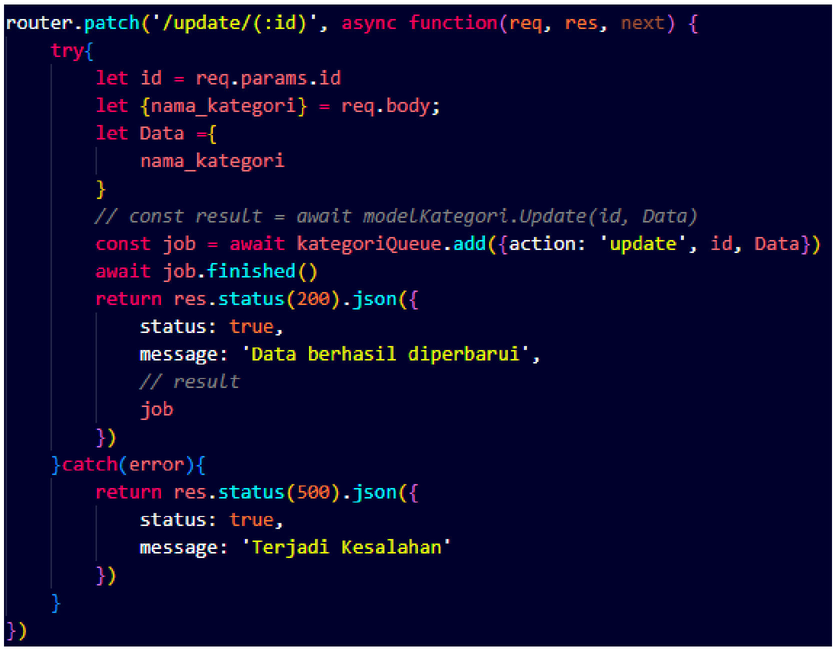
```
const modelKategori = require('../model/modelKategori');
const { kategoriQueue, produkQueue } = require('../config/middleware/queue');
const modelProduk = require('../model/modelProduk');

kategoriQueue.process(async (job) => {
    const { action, id, Data } = job.data;
    console.log(`Memproses antrian kategori... (ID: ${job.id}, Action: ${action})`);

    if (action === 'get') {
        const hasilQuery = await modelKategori.getAll();
        console.log(`Antrian ID ${job.id} selesai: Data kategori diambil.`);
        return { data: hasilQuery };
    }

    if (action === 'store') {
        await modelKategori.store(Data);
        return { message: 'Kategori berhasil ditambahkan' };
    }

    if (action === 'update') {
        console.log(Data);
        await modelKategori.Update(id, Data);
        return { message: `Kategori dengan ID ${id} berhasil diperbarui` };
    }

    if (action === 'delete') {
        await modelKategori.Delete(id);
        return { message: `Kategori dengan ID ${id} berhasil dihapus` };
    }
});
```

Worker berfungsi untuk mengidentifikas action yang kita lakukan sehingga dapat menerapkan queue.

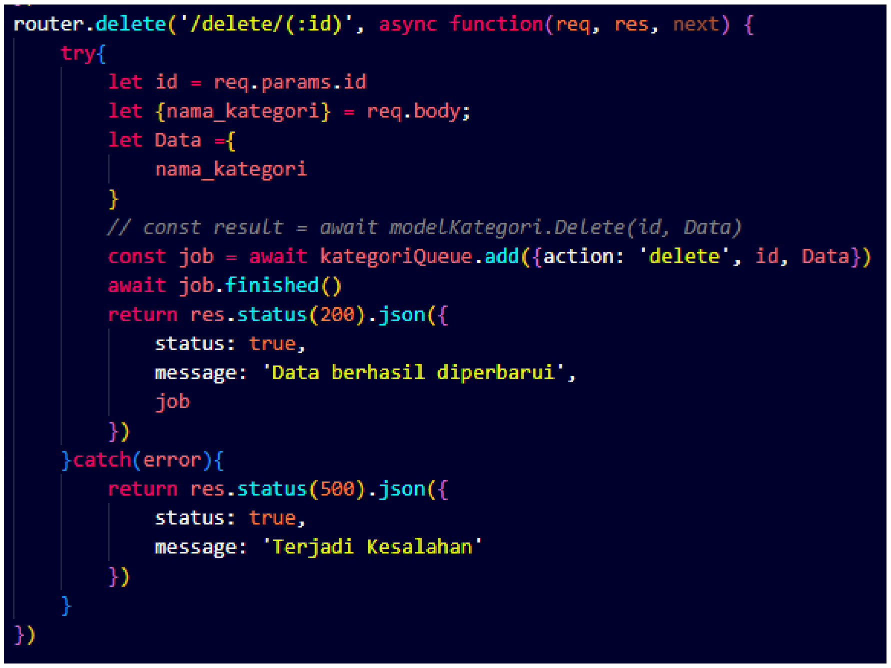
4. Lakukan perubahan dengan mengimplemantasikan pada router get, post, patch, dan delete pada route kategori.



```
router.get('/', cacheMiddleware, async function(req, res, next) {
    const job = await kategoriQueue.add({action: 'get'})
    const result = await job.finished()
    return res.status(200).json({
        status: true,
        message: 'Data Kategori',
        data: result.data
    })

})


router.post('/store', async function(req, res, next) {
    try{
        let {nama_kategori} = req.body;
        let Data ={
            nama_kategori
        }
        // await modelKategori.store(Data);
        const job = await kategoriQueue.add({action: 'store', Data})
        await job.finished()
        return res.status(201).json({
            status: true,
            message: 'Data berhasil ditambahkan'
            // data: result.Data
        })
    }catch(error){
        return res.status(500).json({
            status: true,
            message: 'Terjadi Kesalahan'
```

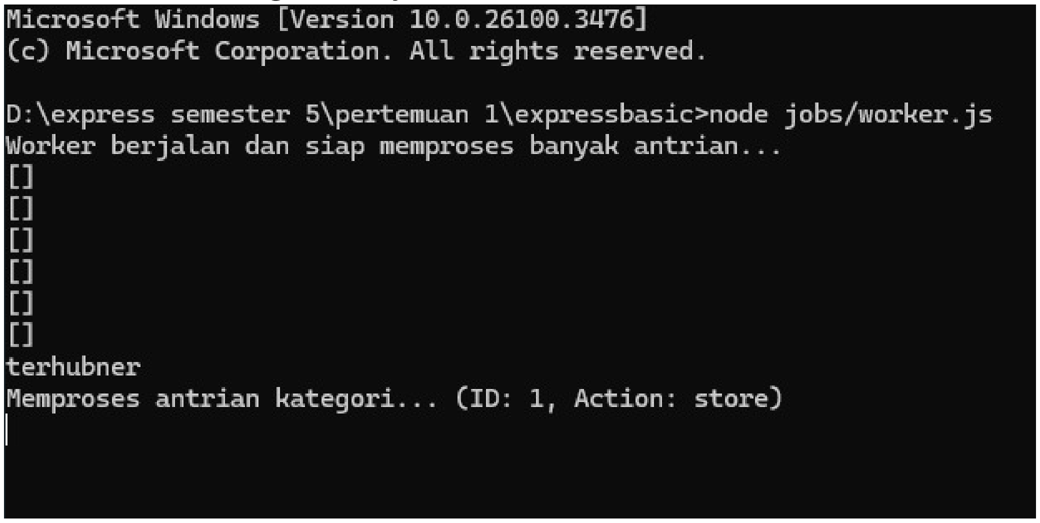


```
router.patch('/update/(:id)', async function(req, res, next) {
    try{
        let id = req.params.id
        let {nama_kategori} = req.body;
        let Data ={
            nama_kategori
        }
        // const result = await modelKategori.Update(id, Data)
        const job = await kategoriQueue.add({action: 'update', id, Data})
        await job.finished()
        return res.status(200).json({
            status: true,
            message: 'Data berhasil diperbarui',
            // result
            job
        })
    }catch(error){
        return res.status(500).json({
            status: true,
            message: 'Terjadi Kesalahan'
        })
    }
})
```



```
router.delete('/delete/(:id)', async function(req, res, next) {
    try{
        let id = req.params.id
        let {nama_kategori} = req.body;
        let Data ={
            nama_kategori
        }
        // const result = await modelKategori.Delete(id, Data)
        const job = await kategoriQueue.add({action: 'delete', id, Data})
        await job.finished()
        return res.status(200).json({
            status: true,
            message: 'Data berhasil diperbarui',
            job
        })
    }catch(error){
        return res.status(500).json({
            status: true,
            message: 'Terjadi Kesalahan'
        })
    }
})
```

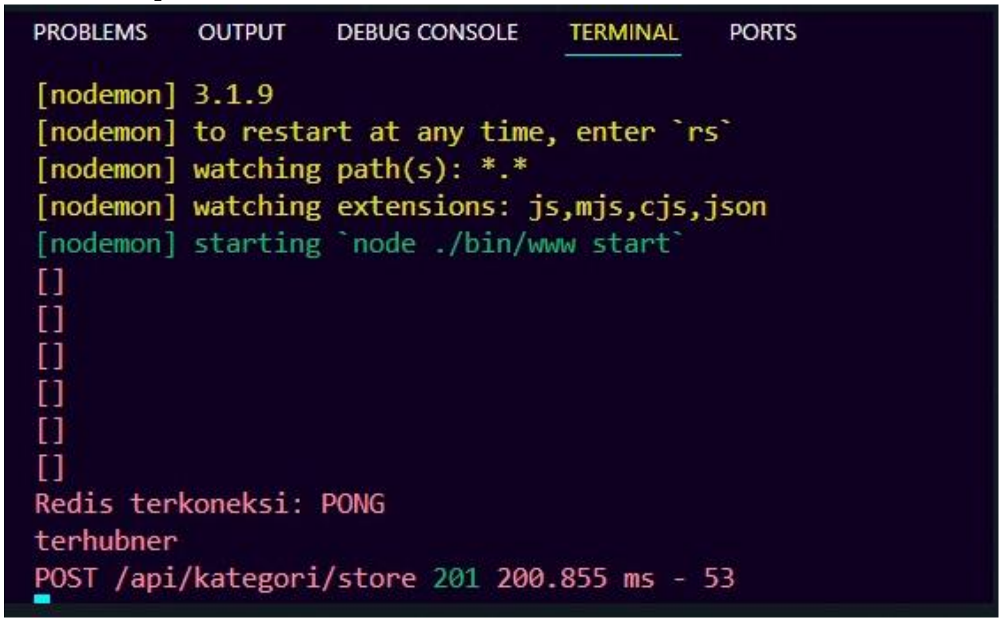
5. Jalankan worker dengan node jobs/worker



```
Microsoft Windows [Version 10.0.26100.3476]
(c) Microsoft Corporation. All rights reserved.

D:\express semester 5\pertemuan 1\expressbasic>node jobs/worker.js
Worker berjalan dan siap memproses banyak antrian...
[]
[]
[]
[]
[]
[]
terhubner
Memproses antrian kategori... (ID: 1, Action: store)
```

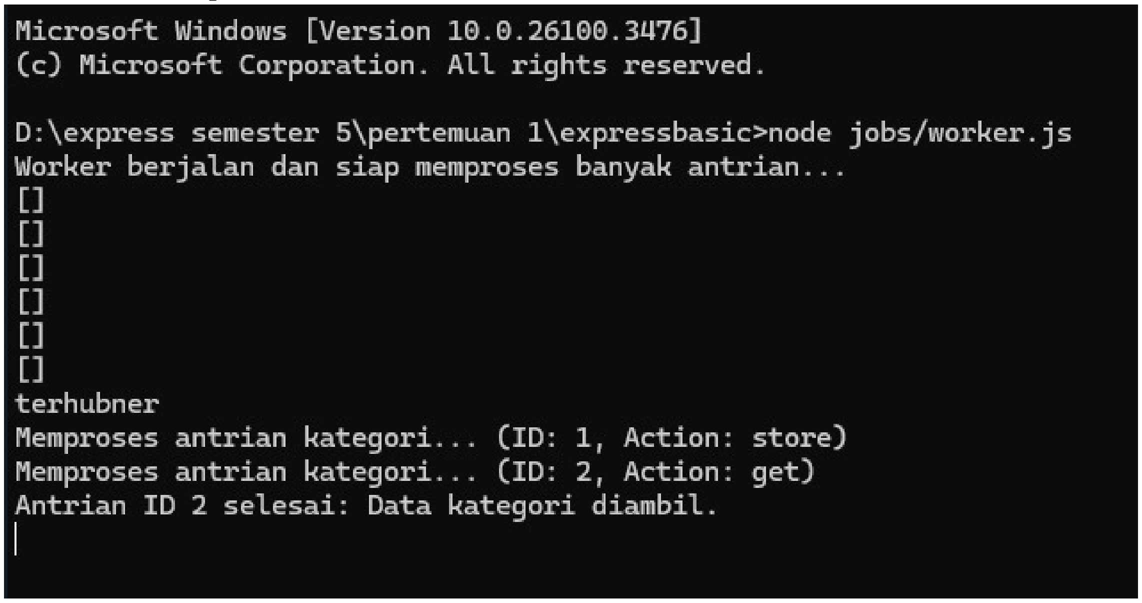
jalankan redis di terminal project yangg berbeda dengan terminal run project

6. Nodemon npm

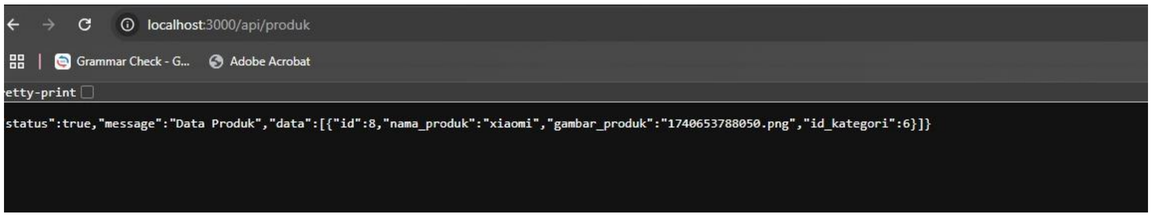


```
[nodemon] 3.1.9
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node ./bin/www start`
[]
[]
[]
[]
[]
[]
Redis terkoneksi: PONG
terhubner
POST /api/kategori/store 201 200.855 ms - 53
```

7. Lakukan beberapa action untuk mencoba



```
Microsoft Windows [Version 10.0.26100.3476]
(c) Microsoft Corporation. All rights reserved.

D:\express semester 5\pertemuan 1\expressbasic>node jobs/worker.js
Worker berjalan dan siap memproses banyak antrian...
[]
[]
[]
[]
[]
[]
terhubner
Memproses antrian kategori... (ID: 1, Action: store)
Memproses antrian kategori... (ID: 2, Action: get)
Antrian ID 2 selesai: Data kategori diambil.
```



```
status":true,"message":"Data Produk","data":[{"id":8,"nama_produk":"xiaomi","gambar_produk":"1740653788050.png","id_kategori":6}]}
```