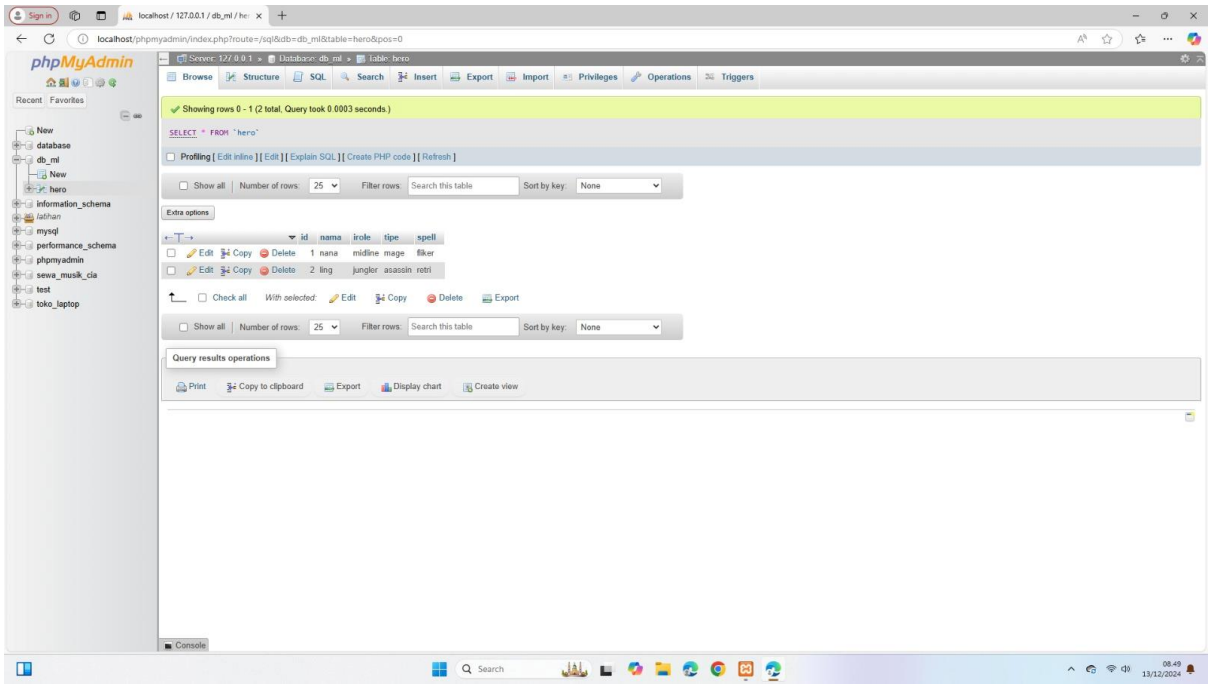
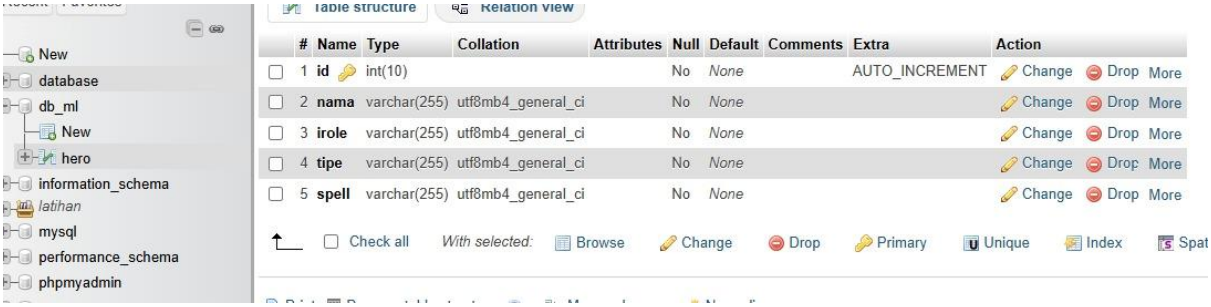
1.masuk ke my php admin untuk membuat databasenya



2.buat datanbasenya contoh untuk database mobile legend yg dibuat adalah id,nama,irole,speii,dan tipe



3. buat API seperti contoh API_rolan kemudian install “body-parser, express, express-validator, mysql, nodemon”

```
{
  "name": "api_rolan",
  "version": "1.0.0",
  "description": "uas",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "rolan",
  "license": "ISC",
  "dependencies": {
    "api_rolan": "file:",
    "body-parser": "^1.20.3",
    "express": "^4.17.1",
    "express-validator": "^7.2.0",
    "mysql": "^2.18.1",
    "nodemon": "^3.1.7"
  }
}
```

4. buat config pada file baru dan masukan databases js sesuaikan dengan databases yang telah dibuat seperti db_ml

```
1
2       let mysql = require('mysql')
3
4       let koneksi = mysql.createConnection({
5           host:'localhost',
6           user:'root',
7           password:'',
8           database:'db_ml'
9       })
10
11      koneksi.connect(function(error){
12          if(!!error){
13              console.log(error)
14          }else{
15              console.log('Connection Success')
16          }
17      })
18
19      module.exports = koneksi
```

5. sesuaikan tabel pada database seperti id,nama,irole,speii,dan tipe jikas sudah selesai masuk ke terminal ketik “npm nodemon start” jika kodingan berjalan maka akan keluar https//local.host.3000, masuk dan copy paste teks nya dan masuk ke web jika status true maka berhasil kemudian masuk ke postman untuk mengupgrade database

```
const express = require('express')
const router = express.Router()

// import database
const koneksi = require('../config/database')
```

```
// insert data & validasi
const {body, validationResult} =require('express-validator')



// membaca data
router.get('/', function(req,res){
   koneksi.query('SELECT * FROM hero ORDER BY id desc',
   function(error,rows){
      if(error){
         return res.status(500).json({
            status: false,
            message: 'database ngga nyambung',
         })
      }else{
         return res.status(200).json({
            status:true,
            message: 'Menampilkan data table hero',
            data:rows
         })
      }
   })
})

//insert data
router.post('/',
   [
      body('nama').notEmpty(),
      body('irole').notEmpty(),
      body('tipe').notEmpty(),
      body('spell').notEmpty(),
   ],(req,res)=>{
      const errors = validationResult(req)
      if(!errors.isEmpty()){
```

```
        return res.status(422).json({errors:errors.array()})
    }

    //mendefinisikan formData
    let formData = {
        nama: req.body.nama,
        irole: req.body.irole,
        tipe: req.body.tipe,
        spell: req.body.spell,
    }

    //masukkan data / query
    koneksi.query('INSERT INTO hero SET ?', formData,
        function(err,rows){
            if(err){
                return res.status(500).json({
                    status: false,
                    message: 'Server mu error',
                })
            }else{
                return res.status(201).json({
                    status: true,
                    message: 'Berhasil input data',
                    data: rows[0]
                })
            }
        }
    )
})

//Detail
router.get('/:id', function(req,res){
    let id = req.params.id
```

```
    koneksi.query(`SELECT * FROM hero WHERE ID=${id}`,
        function(error, rows){
            if(error){
                return res.status(500).json({
                    status:false,
                    message:'Server Error'
                })
            }

            //pencarian
            if(rows.length <= 0){
                return res.status(404).json({
                    status: false,
                    message: 'Data tidak ada'
                })
            } else {
                return res.status(200).json({
                    status: true,
                    message: 'menampilkan data hero',
                    data: rows[0],
                })
            }
        }
    )

})

// Update
router.patch('/update/:id',[
    //validasi
    body('nama').notEmpty(),
    body('irole'). notEmpty(),
    body('tipe'). notEmpty(),
    body('spell'). notEmpty(),
```

```
],(req,res)=>{
    const errors = validationResult (req)
    if(!errors.isEmpty()){
        return res.status(442).json({
            errors:errors.array()
        })
    }

    //id
    let id = req.params.id

    //data post
    let formData={
        nama: req.body.nama,
        irole: req.body.konten,
        tipe: req.body.konten,
        spell: req.body.konten,



    }

    // update query
    koneksi.query(`UPDATE hero set ? WHERE id=${id}`,
      formData,function(error,rows){
        if(error){
            return res.status(500).json({
                status: false,
                message: 'server error',
            })
        } else {
            return res.status(200).json({
                status: true,
                message: 'Berhasil update data'
```

```
            })
        }
       }
    )
})

//Delete
router.delete('/delete/(:id)',
    function(req,res){
        let id = req.params.id

        koneksi.query(`DELETE FROM hero WHERE id=${id}`,
            function(error,rows){
                if(error) {
                    return res.status(500).json({
                        status: false,
                        message: 'Server error'
                    })
                } else {
                    return res.status(200).json({
                        status: true,
                        message: 'data sudah dihapus'
                    })
                }
            }
        )
    })

module.exports = router
```